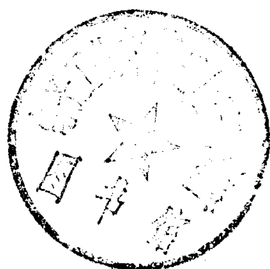
APPLE界面實驗

陳金追 編譯

三葉出版社

APPLE界面實驗

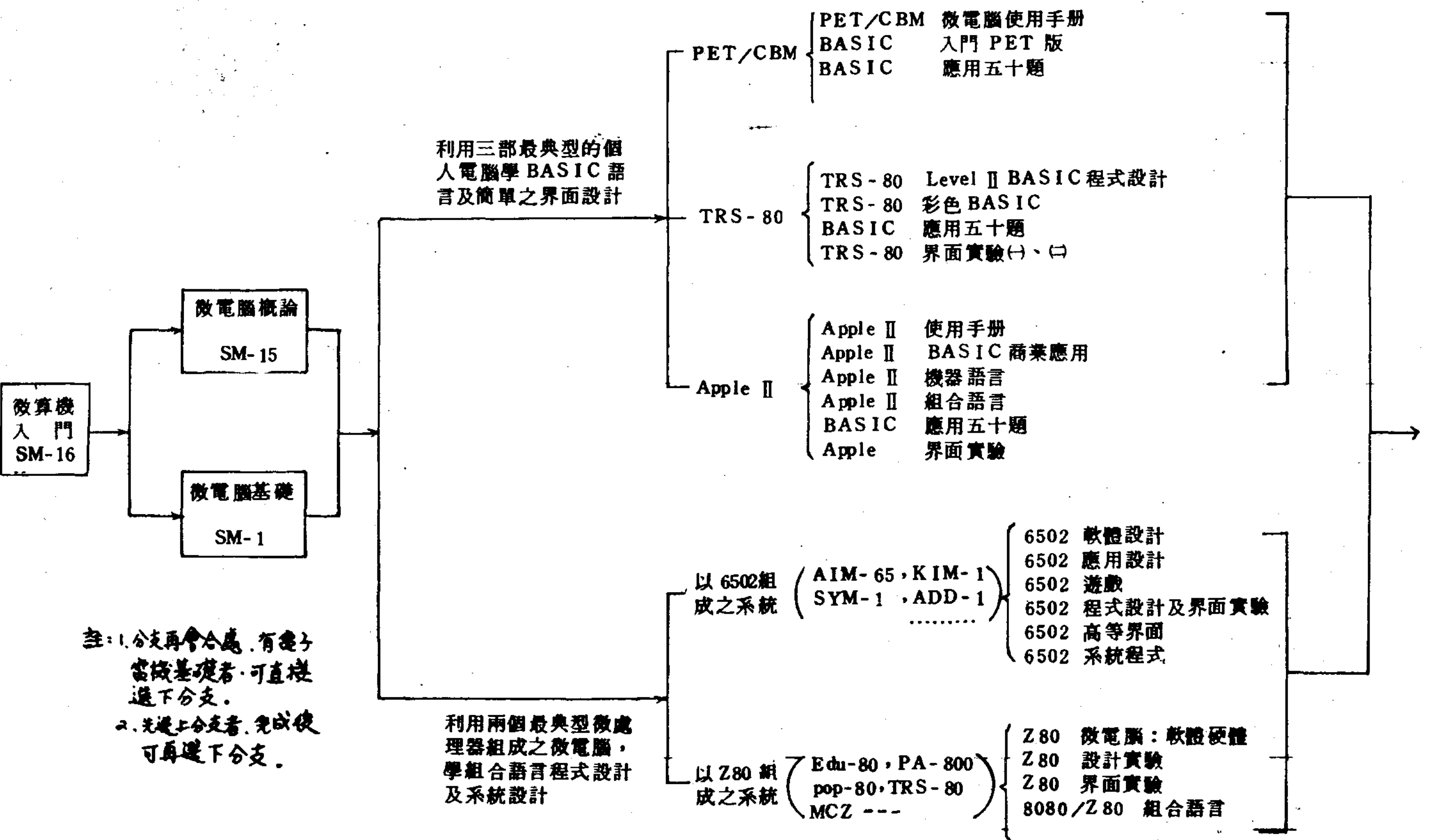
陳金追　編譯





三葉出版社印行

儒林微電腦叢書系統圖（陳金追提供）



序

Apple 就像一股旋風，席捲了美國，吹遍了世界。轉眼，"Apple Ⅱ"一詞已變成世界各地，無論婦孺老幼均耳熟能詳的"字彙"了。微電腦的"威力"，實在無與倫比！

微電腦的最後學習階段應是界面——了解各種輸入／輸出設備如何接至計算機，與計算機達成溝通。這本書將微電腦界面的學習又向前提前了一步，其配合BASIC語言，介紹微電腦一般最基本之界面原理，包括輸入／輸出口之構成、應用，旗號電路之構成、應用，設備位址解碼，以及輸入／輸出指令之用法等。並透過實驗，讓您親眼見證這些原理，使讀者能真正了解微電腦界面的意義，進而自行設計一些簡單的界面電路。全書所談內容均相當淺顯實際，只要對BASIC語言及SN7400系列電子元件略有認識者，均可了解消化並使用這本書。讀完這本書，您會對微電腦界面有一基本且清楚的認識，因此，其不愧是一本非常理想的界面入門書，亦是使用Apple計算機者所不可或缺的良伴。

這本書以Titus，Larsen，以及Titus三人合著的"Apple Interfacing"爲藍本，加上第八章及附錄F編譯而成。第八章所談均爲微電腦界面上經常碰到的實際問題，附錄F則有關Apple計算機系統之擴充。這些材料分別取材自"TRS-80 Interfacing，Vol.2"以及有關雜誌。但願它對您有

所助益。只要有新的資料，本書將隨時搜獵併入。亦盼讀者專家先進，不吝指正。

編譯者

陳　金　追　謹誌

1982.7.1 於台北

前 言

這本書的目的是在向您介紹Apple Ⅱ(註1)計算機內的信號，以及告訴您如何在BASIC語言程式的控制下，以這些信號去控制外部設備。為了加速您電路設計與測試的速度，使您能輕易完成本書所含之各種有趣實驗，我們特地設計了一個一般用途的計算機界面麵包板。透過這個設計系統的使用，您即可專注於相關原理的了解，而不必將時間浪費於檢修電路上。不過，您會有機會製作與測試許多數位電路，以及使用類比至數位與數位至類比轉換器的電路。

本書選用了具有16K可讀寫記憶器的Apple Ⅱ 計算機以及 Applesoft (註2) BASIC解釋程式作例子。這套軟體程式具有相當大的彈性，當您在使用外部界面電路時，您最好擁有它。 Applesoft BASIC解釋程式具有兩個一般用途，可用以傳遞出入計算機之資訊的指令。這兩個指令很容易了解，讀者不必事先詳細了解有關6502 微處理器——Apple的"心臟"——的種種。

註1：Apple與Apple Ⅱ均為Apple計算機公司的註册商標。

註2： Applesoft 為Apple計算機公司的註册商標。

書中首先介紹Apple計算機上可用於界面的控制信號，以及說明這些信號如何使用。有些信號由於並不用於平常之界面電路，而僅用於特殊的界面電路，因此，我們並不介紹。

其次，這本書將告訴您Apple 如何使用兩個一般用途的指令—— PEEK與POKE ——識別或選取外部設備。這兩個指令主要用於控制外部設備；書中除了介紹這兩個命令的動作情形外，更介紹了可用以選取特定輸入／輸出設備之各種電路的用法。您亦可看到Apple 如何經由雙向的資料巴士，將資訊送入與送出外部設備；用作輸入口與輸出口的基本電路在文中有非常詳細的討論。書中亦提供實際的電路，使您能迅速使用這許多例子，設計您自已的界面設備。

您亦可看出BASIC 語言程式的功力——當資料在計算機內部處理，產生有用的結果時。書中以一些簡單的控制程式，告訴您BASIC 語言程式與輸入／輸出設備如何發生關係。讀完之後，您將能自行設計簡單的控制與資料處理程式，控制您的輸入／輸出口與設備。

由於計算機並不一定永遠與外部設備保持同步，因此，計算機與每個輸入／輸出設備之間必須有某種交互作用，以使彼此知道對方是否已可以開始某種作業。這就引發了旗號的問題——計算機與外部輸入／輸出設備之間即以這種信號使資訊能井然有序地傳輸。由於旗號很重要，因此，我們花了相當時間討論這些旗號以及實際用於外部設備內的相對電路。由於旗號一定要控制程式的察覺才有用，因此，書中同時亦討論了軟體。

本書假設讀者對Applesoft BASIC 之指令已有相當的

了解。萬一您正起步開始在學Apple，那作者希望您還是先花一點時間，將FOR，GOTO，IF……THEN，PRINT與INPUT 等幾個指令弄熟之後，再來看這本書。除了這些述句外，其它述句在本書之課文以及實驗內亦均有詳細的討論。等到唸完這本書，所有這些述句的應用對您而言應是易如反掌的事了。

第6章特別提供了 16 個逐步細步說明的實驗，讓您印證在本書前面幾章所學到的一些界面原理。做完這些實驗後，相信您對這些原理會有更深入一層的理解。您同時亦可了解到BASIC 語言程式在作界面控制，以及在實際處理傳入與傳出輸入／輸出設備之資訊上的威力。作者已盡力收獵各種有趣的應用實例。透過實驗，您可明白適用於所有界面電路的基本原理，由最簡單的，到最難的都有。

作者深切體會，欲寫一本對由初學者以至非常有經驗之老手等每一個人均很適合的書實在很難。因此，我們選擇由比初學稍更難一點之處開始談起。因而，本書就忽略了諸如二進數目系統，十進對二進轉換，基本數位電子學，以及麵包板等主題。這些主意在其它許多書上都輕易可以找得到。萬一您您還不熟，那請您隨便選一本有關的書看一看。不過，在適當的地方，我們會特別加上一兩段複習材料，先讓您具備一點基礎之後，再進一步作討論。

本書亦假設讀者對諸如SN7402四倍NOR閘與SN7475四倍鎖住器等之SN7400系列的數位積體電路亦熟悉。至於其它的積體電路，本書則都會或多或少先作點介紹，以讓您有辦法在課文或實驗中使用。若您想再將這些電路用於其它

應用，那作者希望您能進一步參考這些電路之製造商所提供的資料手册。這些資料手册上會有詳細的資料；告訴您電路的各種用法，以及電路的各種變化與特色。

Apple Ⅱ 計算機在其外殼上有八個一般用途的 50 號界面接點。實驗中所使用之基本巴士信號均得自這些接點上的信號，因此，若您決定自行設計與製作一些自己的界面電路，將來插入這些其中之一〝凹槽〞，那您會發現，這些信號在這些邊緣接點上都已經有了。不過，Apple 亦產生一些能簡易界面工作的特殊信號。這些信號與其用法在第 7 章均有詳細介紹。由於這些信號並非是一般性的，其僅爲Apple所特有，而且經常是某一特定接點所特有，因此，其留在最後才介紹。爲了告訴您這些信號如何使用，書中介紹了一個簡單的非同步串行通信界面電路，並列出控制這個電路的軟體。這種界面可用以與其它計算機，串行印字機，模變器（modem），與其它使用非同步串行資料格式的界面元件溝通。

本書並未介紹組合語言程式設計，因爲，這算是一個特殊的主題，而且需要較深的基礎。不過，我們却寫了一個很簡單的組合語言副程式，讓您在許多個實驗中使用。寫這個副程式的原因是，其功能無法以Applesoft達成。這個運算就是八位元 位元組的邏輯AND 運算。在Applesoft裡，邏輯AND運算就是一個結果爲眞或假的運算，其無法簡單地做位元的 AND 運算。這個組合語言副程式同時亦告訴您BASIC 語言程式如何存取諸如此類的副程式。我們選擇使用較複雜的USR(×)命令，而不採用CALL命令，主要是因

為這樣讀者可以學得更多。

我們發現Apple 亦有些限制。舉個例子而言，其即無"四捨五入"指令，可將一個數目四捨五入至某一特定位數。譬如，將 4.1986 四捨五入成 4.20 。同時，Apple亦無位元對位元的 AND 指令，致使這種運算必須藉用一組合語言副程式達成。此外，我們亦發現，測試個別位元的WAIT指令，若條件無法滿足，指令將使計算機"吊死"。除非您按Reset鍵，否則，計算機將一直繼續等到所述條件滿足為止。雖然Apple計算機可做彩色優雅的圖形顯示，但這本書我們一直使用黑白的電視顯示幕。

這本書的目的主要是讓您有個開始，而不是想詳述Apple計算機的所有界面，因此，絕大多數特殊用途之晶片，如類比轉換器，都是選擇最簡單、便宜、而且容易買得到的。這並非指唯有這些產品是擇。當您的經驗逐漸累增之際，您會發現，其它的特殊用途元件仍可達成相同的功能，只不過特色增加，解法更多，或電源不同罷了！

若您對一些較深的主題有興趣，我們建議您讀讀下面幾本書：

6502　軟體設計

6502　程式設計及界面實驗

微電腦類比轉換器：軟體硬體界面。

以上三本書，台北儒林圖書公司均出有中譯本。

作者同時建議您參考一下

TRS－80　界面實驗，第二冊

一書。這本書的抬頭寫的雖然是TRS－80 ，但其所談到有

關高電壓／高電流推動，串行溝通，遙端控制、類比轉換器、濾波器、資料處理，以及一些其它有關的主題等，都是相當一般性而且適用於任何其它機器的。這些主題都是平常您在作微處理器界面時所最容易碰到的問題。很快您會發現，TRS－80 計算機事實上與Apple 計算機小異而大同，而且，兩部計算機的控制信號與 BASIC 指令幾乎完全一樣。

除非特別聲明，否則，本書所例舉之 IC 接腳圖，均以德州儀器（TI）公司所提供之資料為準。此外，Apple與Applesoft 乃美國Apple 計算機公司之註册商標，TRS－80 乃美國Radio Shack 公司之註册商標。

我們希望您能喜歡這本書，從中獲益。亦希望它是您一個好的開始，引導您設計與製作自己的界面電路。

目　　錄

第3章　輸入/輸出界面

第4章　旗號與決策

第5章　Apple之麵包板

第6章　Apple界面實驗

第7章 再談巴士

第8章 馬達、燈炮、電鈴與汽笛

1

6502 微處理器

美國Apple計算機公司所生產的Apple Ⅱ（Apple）計算機系統，乃以 6502 微處理器組成的。這個微處理器晶片構成了這部計算機之中央處理單元（CPU）的心臟，一切算術、邏輯、決策、與其它運算均發生於此。6502 微處理器乃是MOS Technology（MOS技術）、Rockwell International（洛克威爾國際）、以及Synertek（辛勒特克）等幾家公司所製造。

6502 是一部八位元的處理器。因此，所有數學、邏輯、資料傳輸、輸入與輸出等運算均一次八位元地進行。當然，每一位元均可爲邏輯 0 或邏輯 1。6502 使用八位元的資料巴士（data bus），傳遞來回於其本身與各記憶位置以及諸如鍵盤、印字機等輸入／輸出設備間的資訊。若欲傳遞之資訊超過八位元，則資訊就必須先分成許多八位元的片段（或稱字組），然後每次八位元地傳輸。每一八位元的字組通常稱爲一個位元組（byte）。

您應該知道八位元所能表示的最大數值爲 11111111_2 或 255_{10}。因此，當計算機系統所欲運算的數目超過此一數

值時，我們就必須做多位元組的運算。多位元組運算的意義就是，將欲運算的資料以八位元爲單位分段，然後首先運算最低次的位元組，完後再較高一次的位元組，再更高一次的位元組，如此類推，直至整個資料字組均運算完成爲止。依此，八位元的計算機即能處理 255 以上的數值。特別記住，雖然這樣，但 Apple 的 CPU 每次還是僅能傳遞與處理八位元的資料。

6502 以八支接脚與計算機內的資料巴士作連接。資料巴士的用途是傳遞出入於計算機的資訊。由於資訊能沿兩種不同的方向流動，因此，這種巴士稱爲雙向（ bidirectional ）巴士。雙向巴士就像一條早晨車子可沿某一方向行駛，而傍晚汽車又可反方向行駛的公路。

6502 在積體電路（英文簡稱 IC ）上所產生的控制信號，可以監督與管理內部與外部巴士上的資訊流動，使資訊在一個時間僅沿一個方向流動。這些信號的產生與使用本書稍後都會加以探討。

1－1　記憶器

任何計算機系統均具有記憶器。一般而言，記憶器用以儲存控制計算機作業的程式，以及計算機所欲處理的資料。在 6502 計算機內，記憶器可看成就是一系列的記憶位置，而每一記憶位置能儲存八位元，亦即一個位元組的資訊。計算機之記憶器所擁有的記憶位置，通常是 1024 的整數倍。

在計算機術語上，1024 通常簡稱爲 1 K，其正好等於 2^{10}。

計算機必須有辦法選取某一所想要的記憶位置，方能正確地儲存資料或拿取程式指令。由於 6502 微處理器具有16條位址線輸出，這些位址線每條均可輸出 0 或 1，因此，其最高可輸出 2^{16} = 65536（或稱 64K）個不同記憶位址，選取這麼多個記憶位置。當然，每一記憶位置均可儲存八位元的資訊。一般之八位元微處理器由於均具有 16 條位址線輸出，因此，其最多均能選取 64K 個記憶位置，每一不同位址選取一個不同記憶位置。這 64K 個記憶位址即稱爲微處理器之位址空間（address space）。

十六條位址巴士線通常分別稱爲A0、A1、……A14、A15。其中，A0 代表最低次位元（LSB），而A15 代表最高次位元（MSB）。讀者想必都知道，A0 爲 1 時，其眞正的值就僅代表2，而 A15爲 1 時，其眞正所代表的值即爲 32768。由於 6502 微處理器每次僅能處理八位元的資訊，因此，其十六條位址線經常會分成 A15～A8 以及 A7～A0 兩半（微處理器有時亦會處理位址資訊）。其中，A15～A8 稱爲位址之上半或高半（HI）部，A7～A0稱爲位址之下半或低半（LO）部。許多以 6502 構成的微電腦系統，均將高半位址稱爲頁位址（page address）。因爲，若將記憶器等分成 256 頁，每頁含 256 個記憶位置，則每一個高半位址正好可選取一個不同的記憶頁。位址巴士的用法在後面討論軟體程式以及界面電路時，會有進一步的說明。讀者應記住，位址巴士與資料巴士的最大不同是，位址巴士爲單向性，位址資訊僅能由微處理器流至記憶器以及外

部設備。

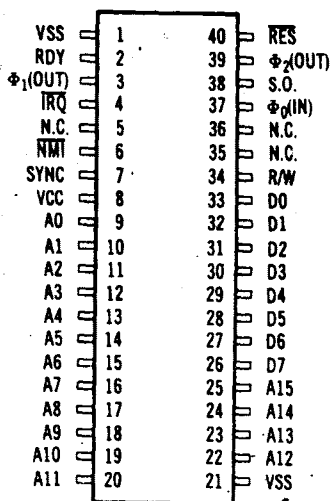


圖1－1　6502微處理器晶片之接脚圖

圖1－1所示即爲6502微處理器之接脚圖。雖然絕大部份信號目前對您而言都無太大意義，但此時您至少應認得8支資料巴士輸入／輸出接脚，以及16支位址輸出接脚。

這一節我們所要討論的是記憶器。一般微電腦所使用的記憶元件有兩種基本型態。這兩種型態是：

1. 可讀寫記憶器（Read/Write Memory，簡記爲R/W M，一般人均慣稱RAM）。可讀寫記憶器主要用以儲存欲改變的程式或資料。微處理器可讀取事先儲存於可讀寫記憶器內之資料，亦可將資料寫入可讀寫記憶器之任一記憶位置。寫入某一記憶位置將恒使該記憶位置原所儲存的資料消失。最便宜的Apple計算機含有16K的讀寫記憶器。

2. 唯讀記憶器（Read-Only Memory，簡記爲ROM）。唯讀記憶器用以儲存永不改變的程式或資料。舉個例子而言，控制整部微電腦作業的監督程式（monitor），或諸如BASIC解釋程式的語言翻譯程式，就都儲存於唯讀記憶器內。微處理器僅能讀取唯讀記憶器內所記憶的資料，而無法將資料寫入唯讀記憶器內的任一記憶位置。儲存於唯讀記憶器內的資料不會因電源之消失而消失。Apple計算機之BASIC解釋程式即儲存於一12K之唯讀記憶器內。

記憶元件還有許多其它更細的分類。譬如說，可讀寫記憶器即有靜態（static）與動態（dynamic）之分。靜態記憶器所儲存之資料會一直記憶至其被改變爲止。但動態記憶器所儲存之資料則必須不斷地"復新"（refresh），否則，其記憶內含將隨時間之過去而消失。Apple計算機系統所含之讀寫記憶器即均爲動態記憶器，其印刷底板上含有必要的復新電路。

唯讀記憶器亦有多種型態，但其一般都是靜態的，主要區別乃在資訊存入記憶位置之方式的不同。唯讀記憶器兩種最重要的型態乃面罩規劃（mask-programmed）式與場規劃（field-programmed）式。面罩規劃式唯讀記憶器的記憶內容是在記憶器製造過程中存進去的，因此，其一般稱爲ROM。場規劃記憶元件的記憶內含則通常是記憶晶片做好後，再以某種特殊的程式規劃電路寫入的（因而，其通常稱爲可規劃唯讀記憶器，英文簡稱爲PROM），並且可在

強烈紫外線的照射下將之抹去，再重新規劃；亦即，寫入新的記憶內含。這種特性在您開發最終欲儲存於唯讀記憶器內的程式時就非常有用。其不需經過耗費昂貴的面罩製作過程，而且程式一有錯，即可立即更正。

有關半導體記憶器最後必須再提示一點。由於電源一拿掉後，記憶內含即消失不見，因此，可讀寫記憶元件屬於揮發性(volatile)元件。另一方面，唯讀記憶器則為非揮發性元件，因為，即使電源拿掉，其所記憶之內含(如BASIC解釋程式)還是永遠存在。

絕大多數記憶器 IC 均沒有接滿 16 條位址線，其皆僅具有足以選取晶片上之所有記憶位置的位址輸入數。因此，一個容量為 64 個位元組的記憶晶片，可能就僅需有6條(因為，$2^6 = 64$)位址輸入線，而容量為 1024 (1K)位元組之記憶晶片，則僅具有10個($2^{10} = 1024$)位址輸入。除了這些輸入外，記憶晶片通常都還具有一額外控制或晶片致能輸入，這個輸入使微處理器一次能選取數個類似晶片。藉著使用適當的解碼與選取電路，我們可以 64 或 1K 位元組的記憶晶片，構成諸如 16K 或 32K 等更大容量的記憶器。這裡主要欲強調的觀點是，雖然一定要組合 16 個位址位元方能選取某一個特定的記憶位元組，但每一記憶晶片上並不必直接接有16 條位址線。因此，當您發現一1K×4位元之記憶晶片僅具有 10 個位址輸入以及一晶片致能輸入時，您應不覺得驚訝。稍後研究輸入/輸出資料傳遞時，這個觀念將進一步澄清。

6502 微處理器產生一個控制信號，負責控制資料巴士

上的資訊流通。這個控制信號即爲讀/寫（READ/$\overline{\text{WRITE}}$，簡稱爲 R/$\overline{\text{W}}$）信號。每次欲作讀取或寫入作業時，6502 微處理器都必須指明一 16 位元位址，以定出與資料傳遞有關之記憶位置。

R/$\overline{\text{W}}$ 信號之W上的短槓，即表示這個信號爲邏輯 0 時爲寫入作業，邏輯 1 時爲讀取作業。由此可見，就僅這一條線即控制了所有的記憶器作業（寫入與讀取）。在若干以 6502 組成的計算機系統與週邊上，您會看到這個信號"分成"記憶器讀取（$\overline{\text{MEMR}}$ 或 $\overline{\text{MR}}$）與記憶器寫入（$\overline{\text{MEMW}}$ 或 $\overline{\text{MW}}$）兩個信號。這種情況必須再將 R/$\overline{\text{W}}$ 信號經過閘控，因此，絕大多數情況均僅使用 R/$\overline{\text{W}}$ 信號本身。R/$\overline{\text{W}}$ 信號在 6502 微處理器晶片上的第 34 脚。

剛講過了，有人會稱讀寫記憶器爲 RAM 。事實上，這是一種錯誤的用法。RAM爲Random-Access Memory 之簡寫，其意義爲"隨機存取記憶器"。實際上，不論唯讀記憶器或可讀寫記憶器，所有的半導體記憶器均算是隨機存取記憶器。微處理器隨時均可直接，並同等迅速地存取記憶器內的任一個記憶位置。而不必像磁帶等記憶媒體，一定要存取過前面的記憶位置後，方能存取到後面的記憶位置。

圖 1 － 2 所示即爲一些典型記憶晶片的接脚圖。有關記憶器，進一步的訊息請您參考下列等項資料：

- *Intel Memory Design Handbook*, Intel Corporation, Santa Clara, CA 95051, 1975.
- *The 8080A/9080A MOS Microprocessor Handbook*, Advanced Micro Devices, Inc., Sunnyvale, CA 94086, 1977.
- *Mostek Memory Products Catalog*, Mostek Corporation, Carrollton, TX 75006, 1977.
- *Bipolar and CMOS Memory Data Book*, Harris Semiconductor Prod. Div., Melbourne, FL 32901, 1978.

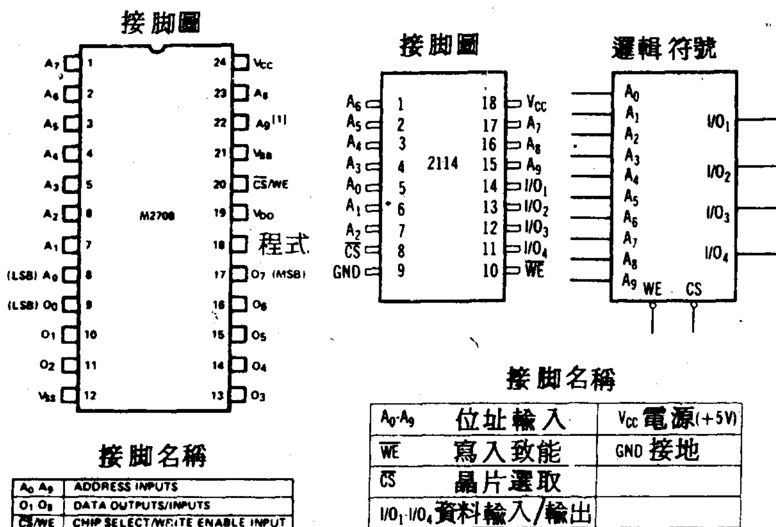


圖1－2　2708 1K×8 RROM與2114 1K×4 R/WM 之接脚圖

1－2　輸入/輸出(I/O)設備

若不接輸入／輸出設備，則絕大多數的微電腦系統幾乎都無什價值。這些輸入／輸出設備可爲諸如讀卡機、印字機、顯示幕等標準週邊，亦可爲察覺器、控制器，與其它絕大部份人都不將之與計算機聯想在一起的元件。Apple亦不例外。其本身亦接有數種輸入／輸出設備：一電視顯示幕，一

卡式錄音機，以及一個鍵盤。

您的計算機亦可加上其它的輸入／輸出設備。這些設備可爲您自己設計的，亦可爲在商場上買到、與Apple相吻合的標準設備。在 6502 微處理器看來，這些輸入／輸出設備極像前一節所討論過的記憶位置。爲了資料能傳入或傳出，因此，輸入／輸出設備連接至資料巴士。此外，爲了能爲 6502 微處理器單獨選取，這些設備亦接至位址巴士上。

R／$\overline{W}$ 控制信號使傳入與傳出輸入／輸出設備的資料流動與 6502 微處理器的作業取得同步。在 6502 組成的計算機系統內，此同一個信號亦用以控制出入記憶器的資訊流動。因此，在 6502 組成的系統內，記憶位址與輸入／輸出設備位址間就無差異。這點在以 8085 或 Z80 微處理器構成的系統就不同，因爲，這兩類型的微處理器將記憶位置與輸入／輸出設備，當成兩種不同的東西看待。換言之，6502（與 6800 ）系統使用的是記憶映像式（ memory-mapped ）輸入／輸出，而 8085 與 Z80 使用的則是隔離式（ isolated ）輸入／輸出。

由於記憶器與輸入／輸出設備均以同一同步信號控制，因此，Apple 之 6502 處理器，任何時刻若不讀取，即爲寫入。當 R／$\overline{W}$ 信號爲邏輯 1 時，即代表 6502 處理器欲自資料巴士讀取資訊；而當 R／$\overline{W}$ 信號爲邏輯 0 時，即表 6502 處理器欲將資訊寫至某一記憶位置或某一外部設備。R／$\overline{W}$ 信號之W上的短橫，即代表此信號爲“邏輯 0 ”時，6502 微處理器進行“寫入”（ Write ）作業。其它信號您可能亦會看到有短橫的情形。這些都代表信號爲邏輯 0 狀態時動作。

由於本書所要討論的全是輸入/輸出設備的用法，因此，有許多東西將留待以後慢慢討論。

複習

至此，您應了解 6502 每次僅能傳遞或運算八位元的資料。更複雜的計算與運算經常則需分數次完成。所有資料均經八位元的資料巴士傳入或傳出 6502 CPU。

表1－1 界面所用的控制信號

資料巴士	D 7～D0	八條雙向的電線，負責傳遞來回微處理器與輸入/輸出設備間的資訊。
位址巴士	A15～A0	十六位元之單向位址巴士，用以選取記憶器與輸入/輸出設備。
	A15～A8	高半位址巴士，最高次八個位址位元。
	A 7～A0	低半位址巴士，最低次八個位址位元。
控制信號	R/$\overline{W}$	讀/寫控制信號。

附註：W上的短橫代表邏輯0爲“動作”狀態。
無論何時，當信號欲轉換成數值時，位元序號愈大，其所代表的數值即愈大。

6502 以一十六位元之位址巴士選取每一記憶位置或輸入／輸出設備。整個位址巴士通常分成高、低兩半，每一半各含八位元。控制信號 R/$\overline{W}$ 控制傳入、傳出 6502 CPU 之資訊流向。這些信號與其記法均如表1－1所示。

1－3　軟體輸入/輸出控制指令

1－3－1　輸入/輸出指令

Apple 計算機具有許多可用以控制輸入／輸出設備的指令。但這些指令絕大部份都用以控制某種特定的輸入／輸出設備或履行某種特定功能。不用介紹，您就已經了解了一些，即令非全部的話。

以下我們舉幾個例子幫您複習一下這些輸入／輸出控制指令。

您最熟的或許是 INPUT 與 PRINT 兩個指令了。其中，INPUT 指令令 BASIC 程式暫停，等候您自鍵盤打入資料。PRINT 指令則將答案或字串“印於”電視螢幕上。

例題1－1　一個簡單的輸入/輸出程式

```
10  INPUT "VALUE OF X IS"; X
20  PRINT " INPUT VALUE WAS"; X
```

執行例題1－1的程式時，除非您已自鍵盤打入變數X

的值，否則，計算機將永遠不會繼續執行程式的第二個指令——20 號之述句。例題中的兩個指令正好可使您自鍵盤打入一個數值，並自電視顯示幕上看到這個數值。INPUT 與 PRINT 述句均有許多變樣，不過，這個例子已足以表明我們所要說的話：您已毫無困難能使用輸入／輸出指令了。

您或許業已發現 BASIC 內亦有圖形顯示的輸入／輸出指令。這些指令諸如HOME、PLOT X,Y以及SCRN(X,Y)。其中，HOME 指令清除整個螢幕，並令閃爍的指標（cursor）回至“原點”位置——螢幕之最左上角位置。PLOT 與 SCRN 則必須以“座標”指明運算所欲發生的螢幕位置。

例題 1－2的程式即舉例說明了圖形顯示指令在一簡短程式內的用法。這個程式在電視螢幕上產生一個位置隨意改變的彩色點。若您用的是黑白電視，您會看到一個濃度一直變化的灰點。

例題 1－2　使用I/O指令之隨機彩色圖案產生程式

```
10  GR
20  X=INT(40*RND(1)) + 1
30  Y=INT(40*RND(1)) + 1
40  COLOR=INT(15*RND(1)) + 1
50  PLOT X,Y
60  GOTO 20
```

另外有兩個指令，您可能一直未把它當成輸入／輸出指令。這兩個指令是用以讀取儲存於卡帶之程式的 LOAD 指令，以及用以將程式存入卡帶的 SAVE 指令。每一個指令均引發一系列事先定義好的運算，控制卡帶機。由於這兩個

指令的用法十分清楚，因此，我們在此就不再另舉例題說明。

其它的輸入／輸出指令則爲與可以鍵盤與電視顯示幕取代之特殊輸入／輸出設備有關的 IN#X 與 PR#X 運算。特別注意，這兩個指令是Apple 與其 BASIC 解釋程式所特有的。其對其它的 6502 計算機系統就無意義，除非這些系統亦使用Apple 的 BASIC 解釋程式。此外，這些指令亦專屬於某一輸入／輸出設備；亦即，HOME 指令對卡式錄音機，或其它任何輸入/輸出設備將毫無影響。同樣地，INPUT 指令亦僅能用於控制自控制鍵盤輸入資料。

1－3－2　通用的輸入/輸出指令

雖然Apple 計算機之 INTEGER BASIC 解釋程式中亦有一些一般用途的輸入／輸出指令，但本書將選用作者認爲更具彈性的APPLESOFT BASIC解釋程式。若您希望將您所用的Apple計算機換成這個程式，則您可與就地的 Apple 經銷商取得連繫。

這兩個輸入／輸出設備指令爲 PEEK 與 POKE 。其中，POKE 將資料自計算機傳遞至某一外部設備，而 PEEK 則將資料自外部設備傳遞至計算機。這些指令使用時均有一定的格式，才能正常動作。

輸入／輸出設備通常稱爲口(ports)。輸出設備稱爲輸出口(output port)，輸入設備稱爲輸入口(input port)。這是全微電腦工業界標準的稱呼法。

輸出指令 POKE 必須指明一資料欲送達之輸入／輸出

設備的位址，以及一欲送給被選取設備之資料值，其實際格式爲

POKE X , Y

其中，X代表欲接收資料值Y之輸出設備的十進位址。資料Y必須爲十進數。由於 6502 微處理器僅能選取65536 個記憶位置，因此，X 之值必須介於 0 與 65535（含）之間。而且，由於計算機僅使用八位元之資料巴士作資訊傳遞，因此，資料值 Y 之範圍必須在 0 至 255（含）之間。舉個例子而言，下面的指令即將數值 215 送至位址 12684 的輸出口。

POKE 12684 , 215

除了不含資料值外，輸入指令 PEEK 完全類似於 POKE 指令。由於指令所欲獲得的即爲呈現於輸入口之數值，因此， PEEK 指令只需指明輸入設備的十進位址就夠了：

PEEK （X）

其中，X爲輸入設備之十進位址。

若僅輸入一數值而不作其它運算，則輸入並無多大意義，因此，輸入指令永遠含在一完整述句中，而不是單獨存在。譬如，下面就是一個例子。

Q = PEEK　(34579)

這個述句將自34579輸入設備所輸入的數值令為變數Q的內含值。記得，PEEK 指令中之輸入設備位址一定要以括弧括起。

PEEK 指令所輸入的數值將恒介於0與255（含）之間。同樣地，這又是八位元傳遞所造成的限制。

表1－2　正確之輸入(PEEK)與輸出(POKE)

指令格式

POKE 45124,98 POKE N,120 POKE 45124,X POKE X,M	L = PEEK (23109) L = PEEK (Q)

輸入與輸出指令內亦可指明變數作口位址或資料值（POKE 指令時）。因此，表1－2所示的所有 PEEK 與 POKE 指令均屬正確。當然，我們均假設在表1－2之指令使用前，N、M、X與Q等變數的值均已事先設定過。

若輸入與輸出指令所指明之資料值超過65535，則Apple計算機將告訴您 ILLEGAL QUANTITY ERROR（量不合錯誤）。試圖輸出一超過255的數值亦會導致同樣的錯誤。

例題1－3的程式即舉例說明 POKE 與 PEEK 指令的用法。雖然這個程式可執行，但由於計算機上並未接外部輸入/輸出口，所以，沒有用。

例題 1－3 PEEK與POKE之簡單輸入/輸出程式

```
10  INPUT "OUTPUT PORT # =";P
20  INPUT "VALUE FOR OUTPUT"; V
30  POKE P,V
40  GOTO 10
```

```
10  INPUT "INPUT PORT # = "; M
20  PRINT "VALUE AT PORT ="; PEEK (M)
30  GOTO 10
```

由於以 6502 組成的計算機並不區分用以暫時儲存程式與資料以及用作輸入/輸出口的記憶位置。因此，我們經常以 PEEK 與 POKE 指令檢查並改變 Apple 內之各種記憶位置的內含。但假若您毫不加辨別地胡亂將資訊 POKE 入可讀寫記憶器，則您可能輕易“蓋掉”程式的某些重要部份，或 BASIC 解釋程式暫時所儲存的資訊，最後導致整個計算機系統的“崩潰”，使您的程式與資料完全喪失或嚴重地改變。所以，在未獲得一些明確的指引前，最好勿隨意地將資訊 POKE 入各個記憶位置。當然，隨時您想要，您都可以 PEEK 指令檢查任一記憶位置的內含，因爲，這個指令執行的結果並不改變被檢視記憶位置的內含。由前面記憶元件的討論，您應知道， POKE 運算對Apple 內之唯讀記憶元件毫無影響。

1－3－3 記憶分佈圖

此時，我們最好看一下Apple計算機所用到之所有記憶位址的使用分佈情形。圖 1－3 所示即爲Apple 計算機上，

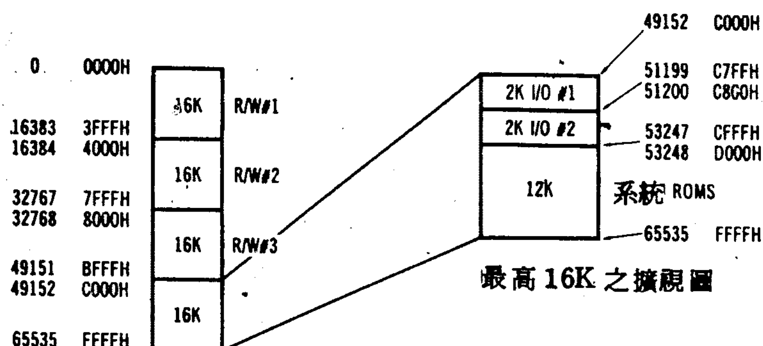


圖1－3　Apple計算機之64K記憶分佈圖

完整 64K 記憶空間的分佈情形。爲了方便起見，所有位址我們均同時寫出十進制與十六進制，並在十六進數後面加一"H"字尾作識別。

Apple 計算機之整個記憶空間（ 64K ）正好分成 4 個各佔 16K 的區段。其中三個區段分配作可讀寫（ R/W ）記憶器。幾乎所有的Apple計算機都接滿R/W #1 的區段，其它的兩個 R/W 區段有時則未接，留作將來擴充用。因爲，絕大多情況下， 16K 的讀寫記憶器都已夠用了。許多廠商都有賣擴加記憶晶片的套件，Apple的用者欲擴接系統的記憶器應不致有困難。

最後的16K區段則留作ROM與輸入/輸出口選取用。Apple的系統ROM，內存 BASIC 解釋程式及監督程式，佔了這個空間的 12K 。所剩的 4K 空間則分成兩個 2K 的空間，作輸入/輸出選取與未來擴充用。其中位址 C000H 至C7FFH 的輸入/輸出區段最重要，因爲，這個區段已特

別騰出作界面用，Apple系統絕不再用作其它用途。這個區段內的某些位址，已經被Apple用作控制揚聲器、鍵盤、與卡式錄音機之類的東西。實際的位址分配如表1－3所示。

表1－3　Apple輸入/輸出位址及其用途

功能	位址	
	十進*	十六進
鍵盤資料	49152	C000
清除鍵盤攫取	49168	C010
揚聲器	49200	C030
卡式錄音機輸出	49184	C020
卡式錄音機輸入	49256	C060
旗號輸入	49249～49251	C061 ～ C063
類比輸入	49252～49255	C064 ～ C067
類比清除	49264	C070
共用攫取	49216	C040

* 僅給正位址。求負位址時，只要加－65536即可。

這些輸入／輸出位址的實際用法，請您參考"Apple Ⅱ 參考手册"與"BASIC 程式設計參考手册"，這些手册都是隨機附送的，要不然，寫信至下列地址亦可買得到。

Apple Computer Inc.

10260 Bandley Dr. Cupertino CA95014

另外的2K區間，C800H～CFFFH，則留作將來擴充用。若您有很長的程式想隨時叫用，則您亦可將這個區段用成ROM。

輸入／輸出位址的實際用途在後面一節會有更詳細的討論。目前，您只要了解已專爲您的應用騰出一段記憶位址就夠了。同時您亦應了解，圖1－3之記憶分佈圖是Apple計算機所特有的。其它 6502 系統的記憶分佈圖可能就不是這樣。其R／W、ROM、與I／O設備位址可能都位於不同的區段上。

1－3－4　軟體指令與界面電路

正如您所知的，不論在輸入／輸出設備或在記憶位置，PEEK 與 POKE 指令均引發某些動作。諸如A＝1.359之指令使一個值存入記憶器內，但我們却不知Apple分派那一個位置給變數A，且不知數值1.359如何儲存。但 PEEK 與 POKE 指令即不同，這兩個指令均引發一系列固定已知的動作：傳送資料位元組，產生控制信號，以及將位址資訊送出於位址巴士線上。這些固定且可再生的動作使我們能以這些指令控制輸入／輸出設備。現在，我們就來探討這每一個軟體指令所引發的動作系列。

PEEK 與 POKE 指令的動作原理非常類似。兩個指令均指明一 16 位元的位址資訊。指令執行時，指令上所含的位址資訊均經位址巴士線A15～A0送給外部設備。如此，所有接至這些位址線上的設備與電路，不論記憶器或輸入／

輸出設備，都將收到此一位址。

執行 POKE 指令時，6502 晶片同時亦輸出指令所含之資料值，但是在資料巴士 D7 ～ D0 上。當資料位元與位址位元均“穩定”或以可用形式呈現於各自巴士上時，6502 即經控制巴士發出READ/$\overline{\text{WRITE}}$ 信號。這個信號同步了被選定輸入／輸出設備的資料獲得。當然，“捕捉”資料，找到被選定之輸入／輸出設備並使其動作與 6502 系統取得同步都必須另加電路。圖 1 － 4 所示即爲這些信號出現於如 Apple之 6502 系統的情形。當然， POKE 指令涉及許多組合語言指令，因此，時序圖中所示僅爲資料傳遞其正發生的片刻。此時此地，我們唯一想知道的僅是 6502 於 POKE 作業時的動作。

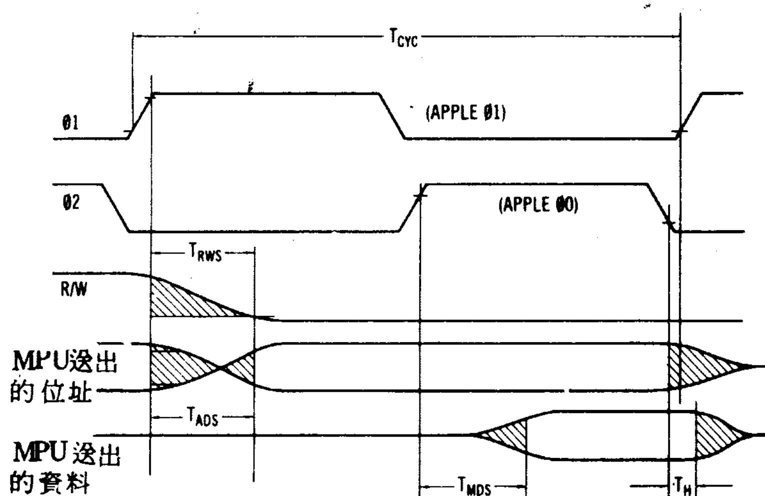


圖 1 － 4　寫入作業之信號關係圖(見附錄)

執行 PEEK 指令時，指令內並不含資料值，而是由外部設備讀取資料值。指令唯獨指明位址。指令執行時，十六位元之位址被置於位址巴士線上。當位址出現時，此一位址所選定的設備即應將其資料置於資料巴士上，以便 6502 微處理器能讀取。讀取作業時，6502 所送出的R/$\overline{W}$控制信號為邏輯1。同樣地，選取輸入/輸出設備以及將設備之資料置於資料巴士上，都必須有額外的電路。 PEEK 指令執行之典型時序圖即如圖 1 － 5 所示。

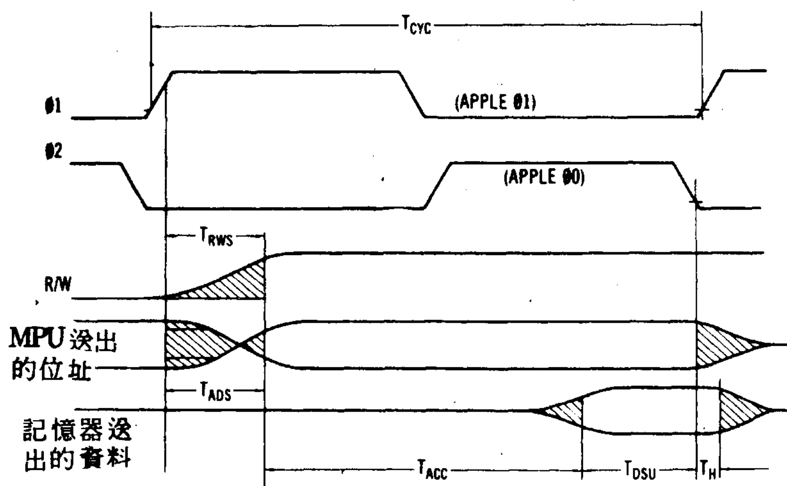


圖 1 －15　讀取作業之信號關係圖(亦見附錄)

接著我們將簡短地介紹一下輸入與輸出口所使用的一些電路。前面我們曾經說過，所謂輸入/輸出口 即為一能接收微電腦所送出之資料，或送出微電腦所欲輸入之資料的元件，而有些 輸入/輸出設備 可能實際含有數個個別的輸入/輸

出口。由於可能要求計算機送出超過八位元以上的資訊，或自己送出八位元以上的資訊給計算機，因此，諸如工業控制器、資料記憶設備（磁碟、卡帶）、類比轉換器、以及一些其它的輸入/輸出設備可能都具有數個輸入/輸出口。但是，不論那一種情況，所有的資料傳輸都必須每次八位元地進行，超過八位元的資訊則必須分批傳送。特別記住：**資訊每次僅能八位元地傳送。**

1－3－5　軟體命令－資料傳遞與控制

絕大多數情況下，PEEK 與 POKE 指令均用以傳遞來回於 6502 計算機與記憶位置或輸入/輸出設備間的資訊。如前面所說的，有時欲傳輸之資料會超過八位元，這時，資料必須分成多個位元組，一次一個位元組地傳輸。

有時，資料傳輸所傳遞之實際值亦有無意義的時候。這時，資料之每一位元均分別代表一種與位元之位置值無關的兩態條件。舉個例子而言，Apple上可接有許多察覺器，分別指示油筒是滿或空，電爐是開或關，冷氣機是開或關等等。這些指示位元的狀態亦可經一八位元的輸入口，以 PEEK 指令輸入至計算機。如此，這時微處理器自輸入口所讀得的數值，如 $100_{10} = 01100100_2$ ，就不是眞正代表一百了。其可能代表目前電冰箱、冷氣機、與刮鬍刀三者均正在使用中，而其它電器品都不用。換言之，輸入資料之每一位元均分別代表一件個別的事件，此一事件僅有兩種可能狀態：成立（邏輯 1 ）與不成立（邏輯 0 ）。譬如，剛剛所舉的例子即

表示其中三個事件成立（三個察覺器為邏輯1），而其餘事件均不成立。

程式亦可根據其它察覺器所測知之狀態，以 PEEK 與 POKE 指令啓動或關閉某一件設備。事實上，Apple所使用的輸入/輸出位址有許多亦指定給諸如揚聲器之簡單開/關設備。因此，下面的簡單指令

```
A = PEEK(49200)
```

即可在Apple之揚聲器上產生一"砰"聲。您該知道，這時候的變數A是一"虛擬"（dummy）變數，其最終值並無意義，因為，這個簡單BASIC述句的主要目的乃在輸出一個脈衝給揚聲器。換言之，其主要乃在與 PEEK 指令"合成"一完整述句。若置於一迴路內，則上述之揚聲器控制指令即可在揚聲器上產生一陣低沈的嗡聲。例題1－4所示即為這種情況。

例題1－4　簡易揚聲器控制程式

```
10 A = PEEK(49200)
20 GOTO 10
```

這裡該記得的重點是，PEEK 與 POKE 指令並不僅單限於控制資料巴士上的資訊傳遞。其亦可用於特定的控制功能，譬如產生一計數器計數所需之脈衝，打開抽風機，或令太陽能收集器傾斜等。

1－3－6 組合語言與BASIC

您在Apple計算機上所設計的BASIC語言程式，與6502 微處理器眞正所能執行的指令間並無多大的關係。正如您所知的，您所設計的BASIC 語言程式，都是經由BASIC解釋程式（ interpreter ）的解釋而執行的，6502微處理器本身並無法直接執行 BASIC 語言述句，其看不懂這種述句。 6502 所能直接執行的，僅是一組事先定義好，功能都非常“小”的機器指令（或組合語言指令）。

舉個例子而言， 6502 微處理器就看不懂 PRINT 指令，因此，其根本無法直接執行下面的指令：

```
PRINT "THIS LOOKS LIKE FUN"
```

而是BASIC 解釋程式看到了 PRINT 運算後，執行了一系列實際將拼出“ THIS LOOKS LIKE FUN ”字串之文數字碼存入顯示記憶器之組合語言指令後，指令才得以執行的。這些組合語言指令均成一串的0和1的形式，其引發了完成 PRINT 運算所必須的一切內部與外部作業。

雖然這本書不談組合語言程式設計，但您應該知道，那是計算機眞正能直接“看得懂”的語言。

爲了完成必要的資訊傳輸，執行 PEEK 或 POKE 指令時， 6502 微處理器都必須執行許多組合語言指令。由於這些BASIC 語言指令都必須經過解釋，即使一個接一個的來

，整個解釋過程都可能變得非常的慢。例題 1－5 所示即為兩個分別寫成 BASIC 語言與組合語言的揚聲器控制程式。這兩個程式所做的事都一樣；在揚聲器上產生一個聲音。只要聽聽兩個程式所產生之聲音的不同，您就可感覺出這兩個程式在執行速度上的差異。

例題 1－5 揚聲器控制之組合語言與BASIC程式比較

BASIC 程式

```
10  A = PEEK(49200)
20  GOTO 10
```

組合語言程式

```
GO      LDY  #$C0
LOOP    LDA  #$0C
        JSR  WAIT
        LDA  SPKR
        DEY
        BNE  LOOP
        JMP  GO
```

執行程式後您會發現，組合語言程式產生一非常悅耳且勻稱的聲音，而 BASIC 程式則產生一低沈的隆隆聲。這個組合語言程式與Apple監督程式所用者類似，其中，WAIT 副程式就是Apple監督程式內一用以產生延遲的副程式。

雖然 BASIC 程式易寫且易除錯，但有時組合語言程式與 BASIC 程式之優越比率竟達 500 比 1 之鉅。組合語言程式設計對初學者而言並不十分適當。

本書主要將使用 BASIC 語言程式設計，甚少提到組合語言程式設計，對 6502 組合語言程式設計有興趣的讀者，請您逕行參考“ **6502軟體設計　陳金追譯　儒林** ”一書。

1－3－7 二進與十進數目系統

Apple 計算機系統所獲得、處理、與印出的都是十進數（基底十），這使其與現代一般人所用的數目系統完全吻合。若計算機所印出的資料值不是十進形式，則相信人們將難於了解與轉換。前面曾經說過了，資料巴士線與位址巴士線均直接接於 6502 微處理器晶片上，因此，這些都是二進制，僅有兩種狀態——邏輯 0 或邏輯 1 。是以，當我們指明 PEEK 或 POKE 指令的輸入／輸出口位址時，我們必須知道，位址（ 0～65535 ）是以二進形式出現於位址巴士上的（ 0000000000000000～1111111111111111）。不論是由十進制轉換成二進制，或反之由二進制轉換成十進制，讀者應該都沒問題。

同樣地， PEEK 與 POKE 指令傳遞出入計算機的資料值，在傳遞過程亦是成八位元二進制形式，每一位元沿一條資料巴士線傳遞。八位元資料巴士乃 6502 晶片之資料處理能力的限制，而非Apple計算機的限制。因此，我們僅能限於作八位元的資料傳遞。這算是一種很大的限制嗎？通常不。因爲，儘管如此，Apple還是能處理大量的資訊（後面您即可看到），並且易於接至輸入／輸出設備。

在結束這章之前，對於位址最後我們必須再提示一點。那就是，Apple計算機的BASIC 解釋程式能同時處理正的與負的位址。這個意思並不是說計算機內眞的有負的位址。您的腦海裡曾經想像過負的街道號碼像什麼樣嗎？在Apple

內有負的位址主要是位址二進等效的原故。舉個例子來說，揚聲器位址49200就相當於－16336。因爲，這兩個數目的十六位元二進表示均完全相同。不過，作者還是建議您多使用正位址，而勿使用負位址。正負位址間的轉換十分容易：(A)將負位址加65536即可獲得其等效正位址，(B)將正位址減65536即可獲得其等效負位址。雖然49200與－16336均產生同樣的十六位元位址，但我們深信，您還是覺得負位址比較抽象且令人迷惑一點。

2

Apple 界面

至此，您可能急欲知道下列幾點：

1. Apple 實際如何將資訊傳至輸入／輸出設備？
2. 輸入／輸出設備實際如何與計算機之運算取得同步？
3. 每一個別的輸入／輸出設備如何辨認或選取？
4. 輸入／輸出設備如何將其資料置於資料巴士上，且其實際如何由資料巴士獲取資料？

由於了解這幾個問題您就能了解到微電腦界面的基礎，因此，這幾個問題十分重要。這些問題將在本章以及其它章回答。爲了加深您的印象，我們亦設計了一些實驗，讓您能親自動手做做。

本章將介紹一些數位電路的例子。作者假設讀者已能看懂邏輯電路圖符號，並且熟悉一些較平常的 SN 7400 系列之TTL 電路。

2－1　輸入/輸出設備之位址解碼

在能開始討論輸入／輸出設備與計算機間之實際資訊傳輸前，我們必須先了解辨認或選取每一個別輸入／輸出設備的電路與信號。當然，選取輸入／輸出設備有許多種方法，我們將探討其中數種方法。不過，由於個別差異與特殊狀況存在，因此，我們並不能每一種方法都作介紹。

當您以兩個一般用途之輸入／輸出命令—— PEEK 與 POKE ——其中之一，寫程式教計算機做資訊傳輸時，6502 處理器一定會產生某些信號，使資料的流動獲得同步。此時此地，我們主要關心的就是位址巴士線的用途。6502 有 16 條位址巴士線可選取記憶位置或輸入／輸出設備。您應還記得，PEEK 或 POKE 指令各自都含有一用以辨認被選定之記憶位置或輸入／輸出設備的十進位址。當然，Apple計算機本身無法區別記憶位置與輸入／輸出口。

2－2　設備選取

計算機所使用的每一輸入／輸出設備都必須能認得其自己的設備位址。由於 PEEK 與 POKE 命令均使用 16 位元之位址，每一輸入／輸出設備必須隨時監聽（注意）著這 16 條位址線（ A15 ～ A0 ），看看有無自己的位址出現（

亦即，自己有無被選取）。輸入/輸出設備之電路有三種方法可監聽位址巴士上是否出現某一個特定位址：

1. 閘控（ gating ）——測知許多邏輯信號的某一種特定組合。
2. 解碼（ decoding ）——這是一種更具彈性的閘控方法，其可測知數個位址。
3. 比較（ comparing ）——將位址巴士信號與一已知位址相比，直至相同為止。

當然，您亦可使用這三種技巧之組合，同時，以上這每一種方法亦皆有許多變化。接著，我們將分別舉例說明以上所列這三種基本位址解碼方法。

2－2－1　以邏輯閘作位址解碼

在以邏輯閘作位址解碼時，位址必須事先已知，您才能知道邏輯閘要如何組合。這裡，我們以設備位址 1010100011110111_2（ 43255_{10} ）作例子。由於二進寫法顯得冗長且繁瑣，因此，您或許覺得寫成十六進制A8F7H較好一點。此外，由於NAND/AND閘乃最基本的邏輯閘，因此，我們將以這種電路組成我們所要的解碼電路。

圖2－1所示即為數種AND/NAND閘的接脚圖，且表2－1為雙輸入 AND 閘與對等 NAND 閘的眞值表。由於諸如SN 7404 之反相器在設備解碼電路上亦很常用，因此，圖2－1與表2－1亦分別列出了這種邏輯閘的晶片接脚

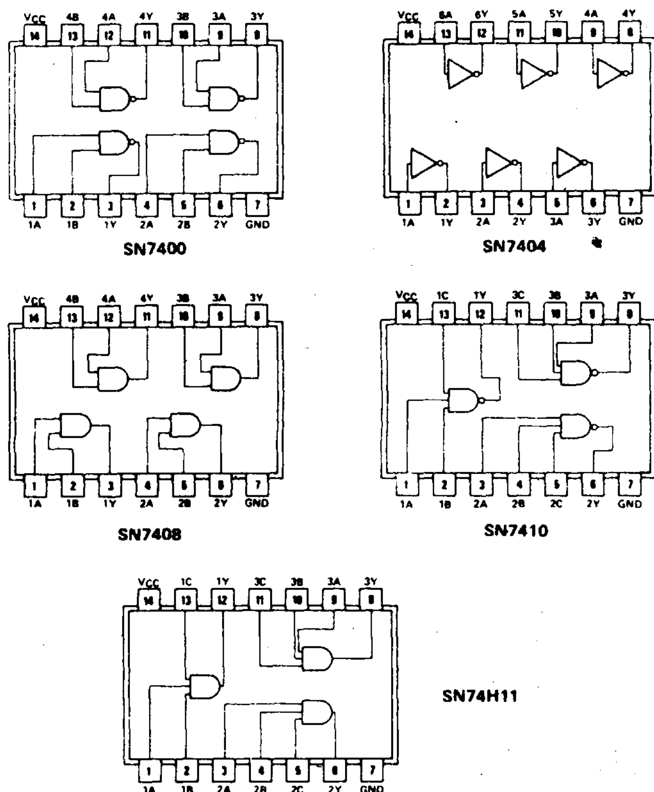


圖2－1 反相器與各種AND/NAND閘的接脚圖

表2－1 雙輸入AND閘與NAND閘，以及反相器的眞值表

AND 閘		NAND 閘		反相器	
輸 入	輸 出	輸 入	輸 出	輸 入	輸 出
A B	Q	A B	Q	A	Q
0 0	0	0 0	1	0	1
0 1	0	0 1	1	1	0
1 0	0	1 0	1		
1 1	1	1 1	0		

圖與眞值表。無論如何，邏輯1均代表高電位（+2.8至+5伏特），而邏輯0均代表低電位（0.0至0.8伏特）。NAND 閘功能有2、3、4、8、與13個輸入等多種，而AND閘則有2、3、與4個輸入等三種。

由於當所有輸入均爲1時，AND閘才輸出邏輯1，NAND 閘才輸出邏輯0，因此，我們必須配組16位元的二進位址 1010100011110111_2，使其在AND或NAND閘的輸入處均能爲1。顯然地，爲達此目的，我們必須令輸入值爲0的位址位元先通過一反相器，然後再加至AND或NAND 閘的輸入。此外，由於沒有16個輸入之AND或NAND 閘，因此，我們使用兩個八輸入的 NAND 閘；其中一個八輸入的 NAND 閘解碼上半位址巴士（A15～A8），另一個解碼下半位址巴士（A7～A0）。然後，這兩個 NAND 閘的輸出結果必須"合"（AND）起來，是以，我們將這兩個八輸入之 NAND 閘的輸出，一起加至一雙輸入的 NAND 閘上。由於 NAND 閘在輸出前將結果反相了，因此，我們分別又在兩個八輸入的 NAND 閘輸出處加了一個反相器。這樣的電路結構即如圖2－2所示。當，而且唯有，所有16個輸入呈 1010100011110111_2 組合時，這個邏輯閘電路才輸出"邏輯0"，否則，其它任何時刻（亦即，16位元輸入之其它組合），此一邏輯閘電路均輸出邏輯1。因此，在將16個輸入分別接至其所對應的位址巴士線後，只要這個邏輯閘電路輸出邏輯0，我們即可馬上知道，現在計算機所送出的位址是 1010100011110111_2，這個設備應被選取。

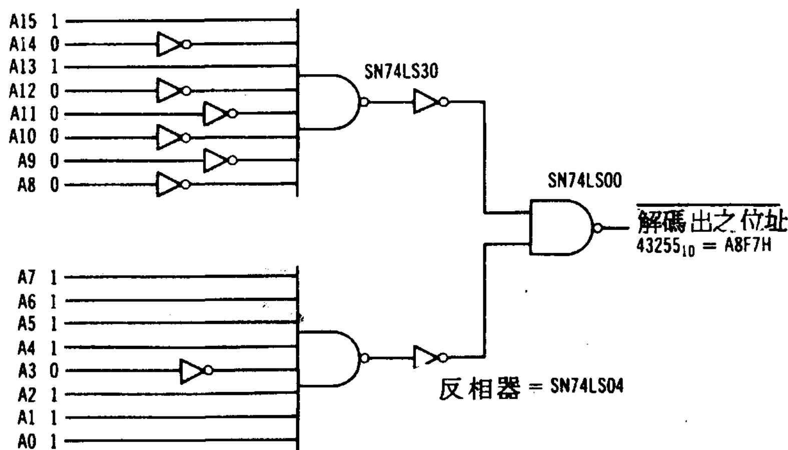
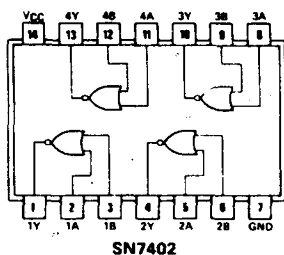


圖 2 — 2　解碼位址43255(＝A8F7H)的邏輯閘電路

這個電路的缺點是，在抵達最後雙輸入之 NAND 閘的輸出前，有些位址信號必須經過四級邏輯閘。由於信號在通過邏輯閘時均會產生時間延遲，因此，這就可能造成某些時序（ timing ）問題。不過，由於這個時間延遲實際上非常小，因此，目前我們將暫時忽略它。倘若延遲時間欲再進一步減少，則兩個八輸入之 NAND 閘的輸出可以一 NOR 或 OR 閘加以組合，而不用 NAND 閘。這種方法非常好。NOR 與 OR 閘到處都可買得到，而且在計算機界面上亦用得很廣泛。圖 2 － 3 所示即為典型 NOR 與 OR 閘的情形，表 2 － 2 所示則為其眞值表。

圖 2 － 2 所示之邏輯電路圖雖然在解碼單一個位址上很有效，而且耗費便宜，但其却沒有彈性。較有彈性的方法可如圖 2 － 4 所示。這個電路圖的優點是，每一位址位元是否



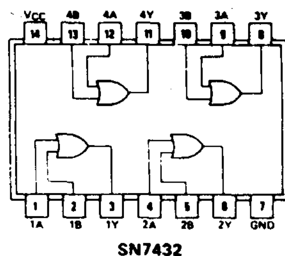


圖2－3　典型NOR與OR閘IC的接脚圖

表2－2　雙輸入NOR閘與OR閘的眞値表

NOR 閘			OR 閘		
輸　入		輸　出	輸　入		輸　出
A	B	Q	A	B	Q
0	0	1	0	0	0
0	1	0	0	1	1
1	0	0	1	0	1
1	1	0	1	1	1

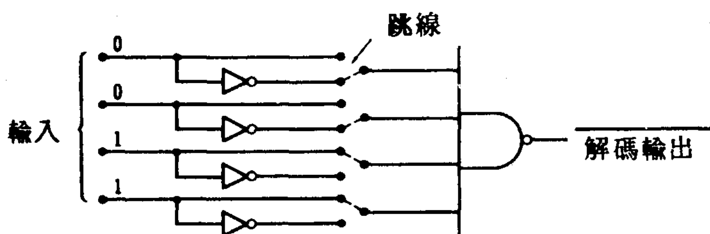


圖2－4　可以解碼多種不同輸入的四輸入邏輯電路

先通過反相器可以選擇。這樣一來，正如圖2－5所示的，整個解碼之邏輯電路就不僅能解碼一個位址。譬如，就圖2－5而言，低次的八個位址位元輸入就可爲01101110_2。若

加上對應的上半部，則這麼樣的一個電路就可解碼65536個可能位址中的任一個位址。（當然，每一次僅能一個位址。）

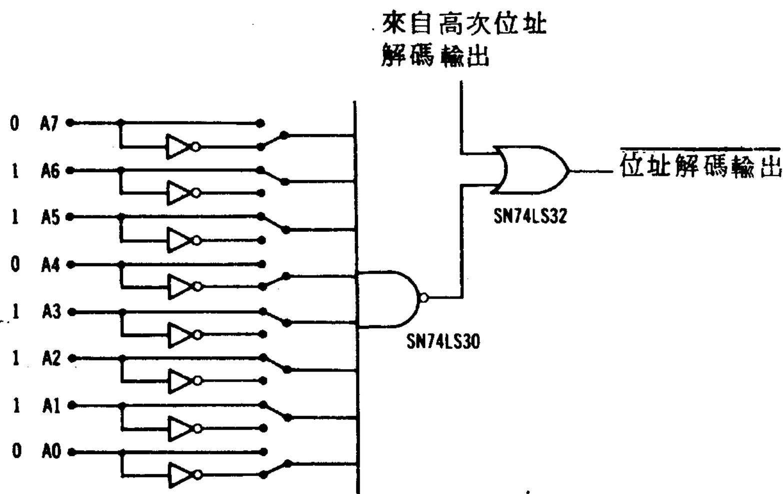


圖2－5 以可規劃的邏輯閘作設備位址解碼（高次位址部份亦同）

這個可規劃的邏輯閘電路就具有相當的彈性，輸入位址可應界面的需求輕易改變，不過，這種電路僅能選取唯一的一個位址，而且這是一個很嚴重的限制。當同一電路板上同時有數個輸入／輸出設備時，每一設備就必須有一各自的位址解碼電路。這種限制可以其它的選取方法加以克服。

很遺憾的，剛剛所示的閘控方式並不足以獨特選取與控制一個輸入／輸出設備。您應記得，在前一章討論讀寫（R／$\overline{W}$）信號時，我們曾說，R／$\overline{W}$信號用以同步來回計算機之資訊流動。因此，若欲正確地使用資料巴士，則輸入／輸

出設備亦須使用這個控制信號。在許多6502計算機系統的界面上，R / $\overline{\text{W}}$線均負責產生一邏輯0的寫入脈衝，而讀取脈衝則另外將R / $\overline{\text{W}}$信號反相獲得。由於兩個都以邏輯0爲動作狀態，因此，這樣所產生的兩個控制信號——$\overline{\text{WRITE}}$（$\overline{\text{WR}}$）與 $\overline{\text{READ}}$（$\overline{\text{RD}}$）——在界面電路上就很容易使用。圖 2－6 所示即爲這兩個信號的用法。在這個電路內，16 位元閘控電路的輸出分別與 $\overline{\text{RD}}$ 及 $\overline{\text{WR}}$ 信號組合，產生兩個控制輸入／輸出口的信號。這兩個控制信號分別是經解碼之位址與 $\overline{\text{WRITE}}$ 脈衝的組合，以及經解碼之位址與 $\overline{\text{READ}}$ 脈衝的組合。這每一個閘所產生的輸出脈衝即稱爲**位址選取脈衝**（address select pulse）或**設備選取脈衝**

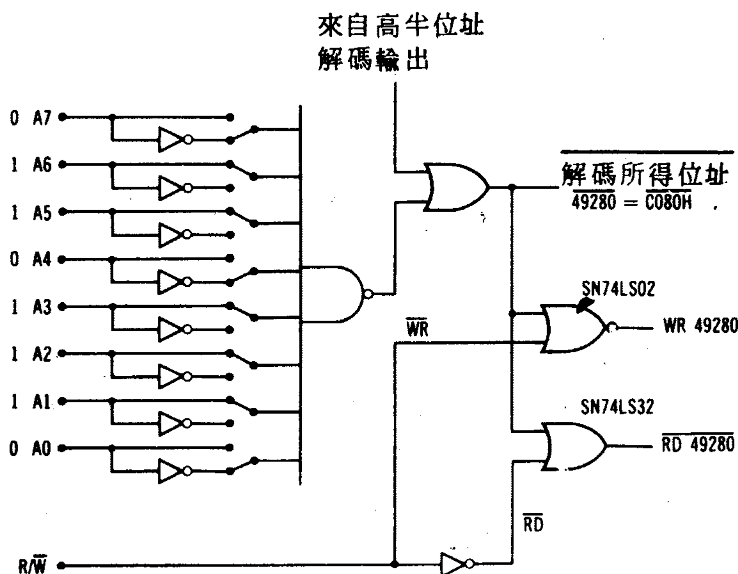


圖 2－6　以$\overline{\text{RD}}$與$\overline{\text{WR}}$信號產生設備選取脈衝，以求得同步。

(device select pulse)。說得更通化一點，經解碼之位址受功能脈衝($\overline{RD}$ 或 $\overline{WR}$)之控制，產生一設備選取脈衝。在圖2－6之電路內，$\overline{RD\ 49280}$ 脈衝即可用以控制一輸入口，而WR 49280 脈衝即可用以控制一輸出口。注意到，WR 49280 上面沒有加一“短横”。因此，這個控制信號是邏輯1時為動作準位，而 $\overline{RD\ 49280}$ 則為邏輯0時動作準位。注意，若寫成十六進制，則這個例子之輸入/輸出口即為 C080H。

在進一步討論之前，您應了解，所謂讀取作業即將資訊自某一輸入口讀入計算機，而寫入作業即將資訊自計算機送出給某一外部設備。同時，以一個位址控制一個輸入口與一個輸出口亦是相當有用而且適當的。這是因為 $\overline{RD}$ 與 $\overline{WR}$ 脈衝永遠不會衝突(任何時刻均僅可能有一者動作)，所以，一輸入口與一輸出口共用同一個位址亦不致造成問題。但是，兩個輸入口就不能共用同一個位址，而且兩個輸出口亦不能。事實上，您可能發現，即使一輸入口與一輸出口共用同一位址，但就功能而言，兩者可能毫不相干，而且可用於不同的界面電路上。

這一節所討論的觀念與基本電路都非常重要，其在其它章節會有更進一步的延伸。您務必要了解我們所討論的這些選取設備的信號。記得，我們一直尚未描述輸入與輸出設備像什麼樣，以及其如何動作，這些都將在下一章討論。

2－2－2 以解碼器作位址解碼

經常，以解碼器（ decoder ）電路取代邏輯閘位址測知電路反而更容易，而且，以解碼器電路取代 NOR 閘設備選取電路有時亦是。為何解碼器這麼好用呢？或許，最好我們應先看看幾種解碼器的樣子以及其如何動作。在研究解碼器時，希望您隨時記得，解碼器只不過是整合在一起，以易於使用的一群邏輯閘罷了。

解碼器電路的規格通常說成X線對Y線解碼器，其中，X代表解碼器之二進輸入數，而Y代表可能的輸出個數，或者X輸入所可能呈現的不同二進組合個數。舉個例子而言，若解碼器具有4個輸入，則由於4個位元可能有 16 種不同的輸出，故這個解碼器即稱為“ 4線對 16 線 ”的解碼器。事實上，後面您會看到，這個規格是一個實際的解碼器。

解碼器的每一個二進輸入都有兩種可能的狀態，亦即，邏輯0與邏輯1。這些輸入之間彼此互不相干。就其僅有兩種可能值而言，輸出亦是二進的，不過，其並不完全獨立。解碼器每次僅有一個獨特（與衆不同之意）的輸出，這個輸出即代表二進輸入的數值或“權重”（ weights ）。經常，這個獨特的輸出均為邏輯0，而其它的輸出則均為邏輯1。

典型的解碼器電路為編號SN74LS139 的積體電路（英文簡稱 IC ）。如圖2－7所示，這塊積體電路（或稱晶片）事實上含有兩個完全獨立的2線對4線解碼器。SN74LS139 的眞值表如表2－3所示。

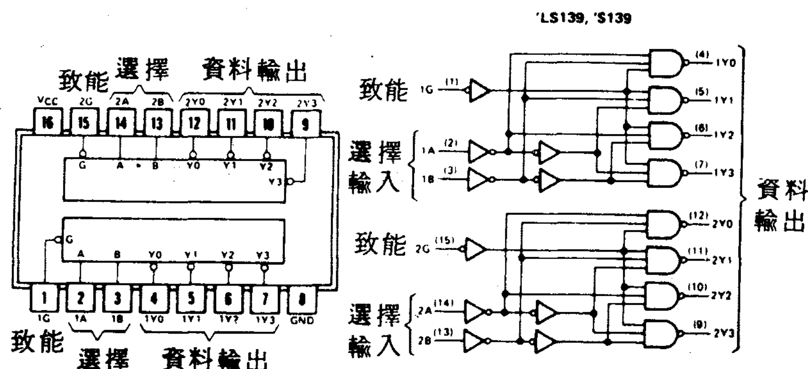


圖2－7　SN74LS139解碼器晶片之電路圖與接腳圖

表2－3　SN74LS139解碼器的眞值表

輸入			輸出			
致能	選擇					
G	B	A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

H＝高電位　L＝低電位　X＝不管

當然，表2－3所示之眞值表對SN74LS139 晶片內所含的兩個解碼器都有效。絕大多數的解碼器電路都含有一致能（ enable ）輸入（或稱動作控制輸入），這個輸入可控制整個解碼器的動作與不動作。就 SN74LS139 而言，這個輸入就是 ENABLE 或“G”輸入。您可發現，當“G”輸入爲邏輯1時，不論輸入A與B如何，所有輸出均被迫成邏輯1狀態。這種情況表示解碼器目前已完全失效。如此

，我們可不必取掉電源而讓整個解碼器失效不動作。

現在，我們來看看一個以 2 線對 4 線解碼器作設備位址解碼的極簡單例子。假設我們僅有少數幾個輸入 / 輸出設備，且 SN74LS139 IC 上的解碼器已足數使用，則圖 2 － 8 所示之解碼器電路即爲一典型的應用例子。這個電路僅解

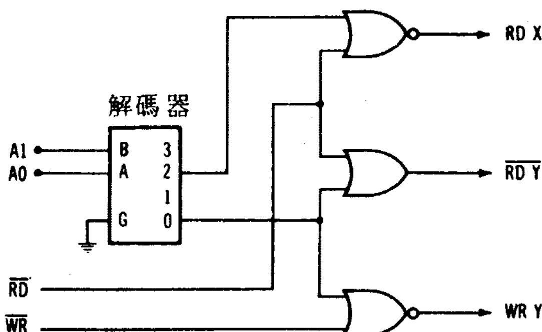


圖 2 － 8 以一 2 線對 4 線解碼器作設備選取

碼位址巴士的其中兩個位址位元，而忽略其它位元。注意，爲使解碼器之輸出能正確動作，致能輸入 G 已接地。另外所增加的 NOR 與 OR 閘產生實際的設備選取脈衝。

由於沒有明確的位址，因此，幾個設備選取信號我們分別稱之爲 RD X、RD Y、與 WR Y。就圖 2 － 8 之結構而言，不論 PEEK 命令所使用的位址是 01010101 00000010、00011101 11110110、或 00000000 11111110〔譬如，A = PEEK (21762) 〕，解碼電路都將產生 RD X 設備選取脈衝，因爲，這幾個位址的最低次兩位元 $A_1 A_0$ 都等於 10_2。由於解碼電路中並未解碼 $A_{15} \sim A_2$ 之位址位元

，因此，這幾個位址位元的內含爲何均無關。這種設備選取的方式即稱爲非絕對（ nonabsolute ）設備選定。所謂非絕對設備選定，即有好幾個位址都可產生選取脈衝。

圖 2 － 8 的電路解碼 4 個位址，因此可選取 8 個不同的設備，包括 4 個輸入設備與 4 個輸出設備。當然，還必須加上其它的 NOR 或 OR 閘。這種解碼方法雖然沒有多大的彈性，輸入／輸出設備的數量不能超過 8 個，但其在小型系統已堪稱夠用。由於其隱含兩個可應用至其它解碼電路的觀念，因此，讓我們再仔細地看看這個電路。

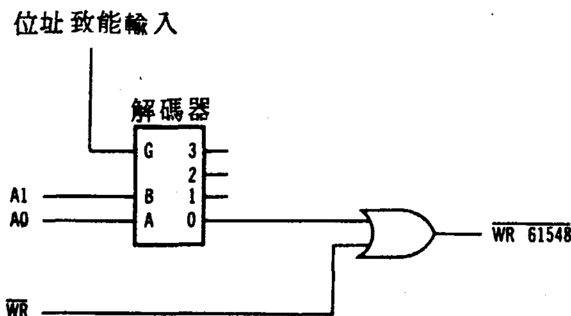


圖 2 － 9　絕對位址選定所用的解碼器

圖 2 － 8 之解碼器的致能輸入“ G ”就直接接地，以使解碼器恒履行解碼功能。這個輸入可使解碼器用於絕對解碼。只要我們使用一個閘控電路，使這個閘控電路唯有在 A15～A 2 等位址線出現某一種位元組合型態時，才供應一個致能信號給解碼器之“ G ”輸入，原來的解碼器電路即可變成絕對選取。而這種閘控電路在前面我們已經討論過了，圖 2

－5 所示的電路就是一個很好的例子。這個電路即可用以產生以上我們所講的致能輸入。由於 A1 與 A0 兩條位址線已用於解碼器中，因此，其就不必再用作閘控電路的輸入了。圖 2－9 所示即爲一個簡單的例子。如圖所示，解碼器之致能輸入來自另一閘控電路。若使用圖 2－5 所示的電路，那 A1 與 A0 的變換開關就不接。

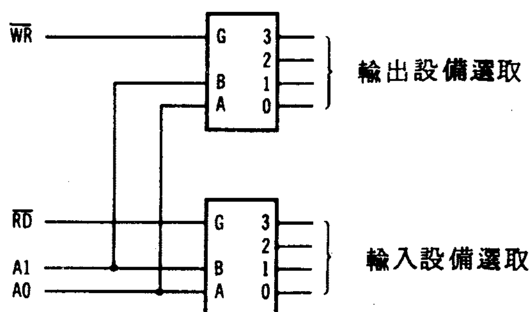


圖 2－10　解碼器之致能輸入與 $\overline{WR}$ 以及 $\overline{RD}$ 合用，產生設備選取信號

假如圖 2－5 所示之電路加上一解碼 1111 0000 的上半位址閘控電路，並且 A1 與 A0 兩條最低次位址線不接，則圖 2－9 之解碼電路將僅有在位址輸入爲 11110000 01101100 至 11110000 01101111 時才能動作。如此一來，這個解碼電路的 0、1、2、3 等四個輸出將分別對應於 61548、61549、61550、與 61551 等四個設備位址。（這四個位址寫成十六進制則爲 F06CH、F06DH、F06EH、與 F06FH。）當然，每一設備選取脈衝的產生都必須加

上額外的 NOR 或 OR 閘（如圖 2 － 9 所示）。

另一種方法是SN74LS139 IC上的兩個解碼器電路都使用，以 $\overline{RD}$ 與 $\overline{WR}$ 兩個功能脈衝推動（致能）解碼器。如此，位址選取又變成了非絕對性，不過，設備選取閘控則在晶片內完成。圖 2－10 所示即爲這種情形。您可發現，產生設備選取脈衝已不再需要 NOR 閘與 OR 閘。雖然這種電路並不一定立即有用，但其却眞正說明了以解碼器之致能輸入產生設備選取脈衝的用法。解碼器之閘控或致能輸入可用以產生設備選取脈衝，亦可用作絕對解碼。有時，其亦可兩種

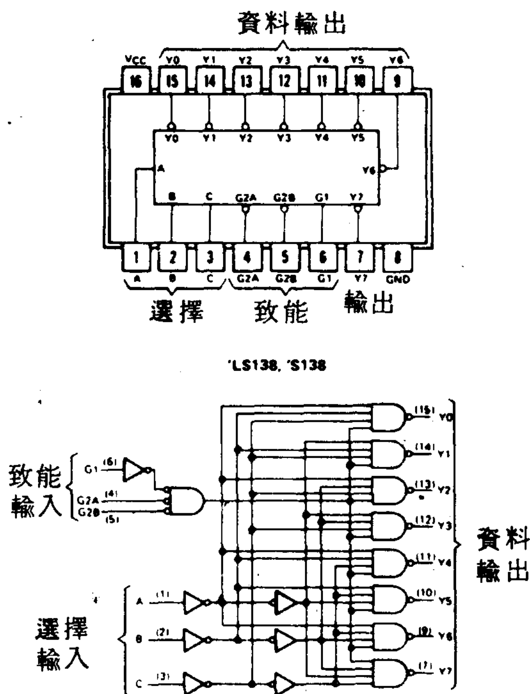


圖 2－11　SN74LS138解碼器

都用。

2－2－3 大型解碼器

在做Apple計算機與外部設備的界面工作時，您可能還會用到一些其它的解碼器電路。視您所選用的型態而定，這

功能方塊圖及輸入與輸出電路圖

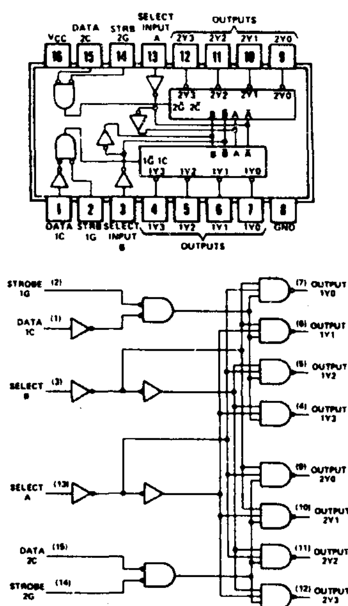
正邏輯：見功能表

功能表

輸入						輸出															
G1	G2	D	C	B	A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	L	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H
L	L	L	H	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H
L	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H
L	L	H	L	L	L	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H
L	L	H	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H
L	L	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H
L	L	H	H	L	L	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H
L	L	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L
L	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	L	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H

H = high level, L = low level, X = irrelevant

圖2－12 SN74154解碼器



功　能　表

2線對4線解碼器

或1線對4線多工器

INPUTS				OUTPUTS			
SELECT		STROBE	DATA	1Y0	1Y1	1Y2	1Y3
B	A	1G	1C				
X	X	H	X	H	H	H	H
L	L	L	H	L	H	H	H
L	H	L	H	H	L	H	H
H	L	L	H	H	H	L	H
H	H	L	H	H	H	H	L
X	X	X	L	H	H	H	H

INPUTS				OUTPUTS			
SELECT		STROBE	DATA	2Y0	2Y1	2Y2	2Y3
B	A	2G	2C				
X	X	H	X	H	H	H	H
L	L	L	L	L	H	H	H
L	H	L	L	H	L	H	H
H	L	L	L	H	H	L	H
H	H	L	L	H	H	H	L
X	X	X	H	H	H	H	H

圖 2－13　SN74155解碼器

些解碼器電路可能還有其它的輸入，致能控制線、以及輸出。圖 2－11 所示的 SN74LS138解碼器與圖 2－12 所示的 SN74154 解碼器就是其中兩個例子。由於具有兩部份，故圖 2－13 所示的 SN74155 解碼器亦算在內。不過，位址輸入，A與 B，則是兩個解碼器部份所共用。 SN74155 的每一部份均有個別的控制或致能輸入。

諸如 SN74154 4 線對 16 線解碼器等之大型解碼器具有相當廣泛的位址解碼彈性。單獨一個 SN74154 解碼器即可用以非絕對地解碼 16 個位址，而當 WR 或 RD 再作爲其中之一致能輸入時， SN74154 又可用以直接產生 16 個設

備選取脈衝，而不需另加閘控。圖 2-14 所示即為這種用法。

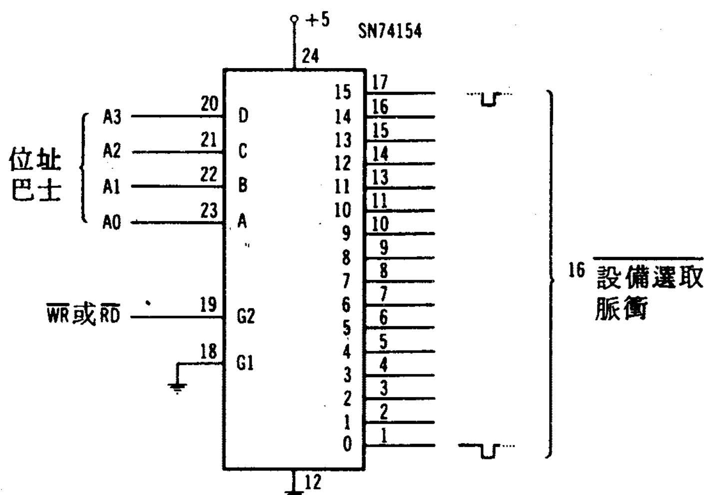


圖 2-14　以SN74154解碼器產生16個非絕對解碼的設備選取脈衝

這個基本電路可再加上額外的解碼器或邏輯閘，進而產生經絕對解碼之設備選取脈衝。圖 2-15 所示即為一個典型例子。$\overline{RD}$ 或 $\overline{WR}$ 信號都可作為下面一個解碼器的閘控或致能輸入。電路上半部所產生的位址選取信號以及下面解碼器所產生的位址選取加功能脈衝，以 NOR 閘組合在一起。因此，上半部電路等於下半部電路的“認可”電路，其使下面解碼器所產生之位址選取變成絕對性。這個圖形同時畫出了兩個設備選取脈衝。這個電路雖然能正常動作，但其並不實用，因為，其可再進一步簡化。

由於 SN74154 解碼器具有兩個致能輸入，G1 與 G2

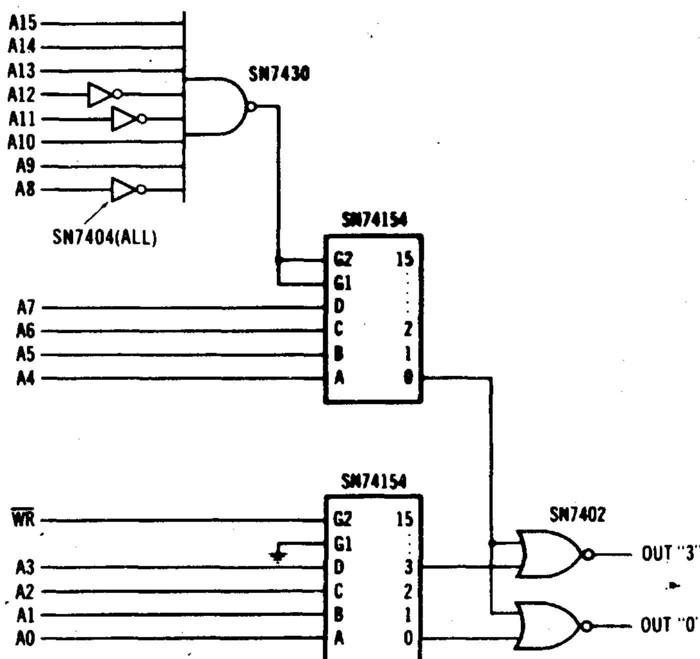


圖 2 —15　以SN74154解碼器與閘控作絕對設備位址選取

，因此，若以第二個致能輸入作爲下半解碼器的“認可”，推動解碼器，圖 2－15 中的 NOR 閘即可免去。圖 2－16 所示即爲這種用法。就這個圖形而言，下面的解碼器現在變成了擁有兩個致能輸入信號；一個爲來自計算機的 $\overline{RD}$ 信號，另一個爲來自電路上半部之致能信號。特別注意，上面解碼器的兩個致能輸入同時使用，因此，唯有高半位址巴士呈現某種特定位元組合時，此一上面解碼器才會動作。如圖所示，A15 ～ A8 等位址線經閘控產生上面解碼器所需的致能信

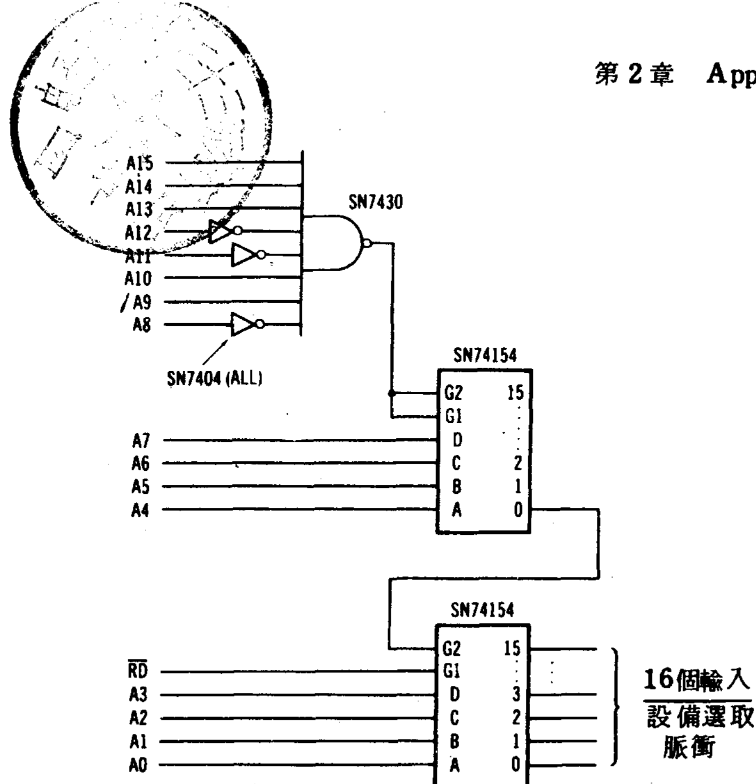


圖 2－16 經改良過的設備選取電路

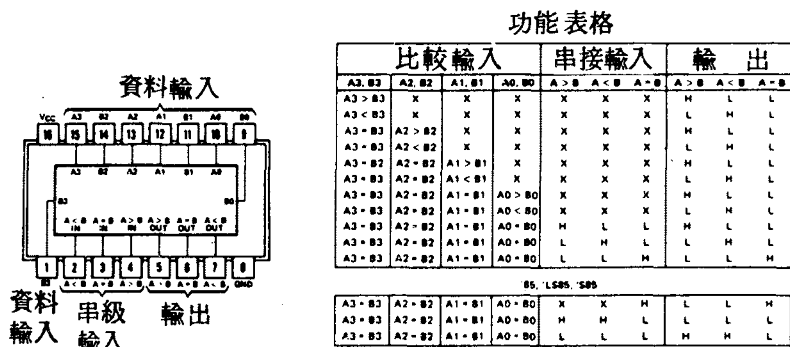
號。

這個電路可再加上第三個解碼器，以產生輸出設備之設備選取脈衝。除了應使用 $\overline{WR}$ 信號而非 $\overline{RD}$ 信號外，這個外加解碼器的輸入與下面一個解碼器的輸入完全相同。

其它還有許多解碼方式同樣可行，稍後在實驗中您就有機會探討各種解碼器的用途。目前我們所要強調的重點是，解碼器的使用簡化了設備選取與閘控的過程。解碼器一般均用於必須具有彈性以及產生數個設備選取或設備位址信號彼此相近之時。

2－2－4　使用比較器

最後我們所要討論的技巧是以數位比較器作設備位址測知。以比較器爲主的方法十分直接了當，其極類似於如圖 2－4 與 2－5 所示的"閘規劃"法。記得，比較器亦只不過



功能表格

比較輸入				串接輸入			輸出		
A3, B3	A2, B2	A1, B1	A0, B0	A > B	A < B	A = B	A > B	A < B	A = B
A3 > B3	X	X	X	X	X	X	H	L	L
A3 < B3	X	X	X	X	X	X	L	H	L
A3 = B3	A2 > B2	X	X	X	X	X	H	L	L
A3 = B3	A2 < B2	X	X	X	X	X	L	H	L
A3 = B2	A2 = B2	A1 > B1	X	X	X	X	H	L	L
A3 = B3	A2 = B2	A1 < B1	X	X	X	X	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 > B0	X	X	X	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 < B0	X	X	X	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	L	L	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	H	L	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	L	H	L	L	H

'85, 'LS85, 'S85

比較輸入				串接輸入			輸出		
A3 = B3	A2 = B2	A1 = B1	A0 = B0	X	X	H	L	L	H
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	H	L	L	L	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	L	L	H	H	L

圖 2－17　SN7485四位元大小比較器晶片

是一群彼此連接在一塊兒，以達成比較功能的邏輯閘罷了！比較器電路使我們能呈現一個不斷與位址巴士上的 16 位元值相比的位址。比較作業由比較器晶片上的閘控電路完成。典型的比較器如圖 2－17 所示的 SN7485 四位元大小比較器。除了對等情況外，SN7485 亦能測出大於與小於的狀況，不過，這兩種情況在位址比較上不用。留意：**SN74L85** 型的 **SN7485** 接脚並不完全相同。詳情請參閱製造商所提供的資料說明書。

圖 2－18 所示即爲一典型的位址比較法。爲了清楚起見

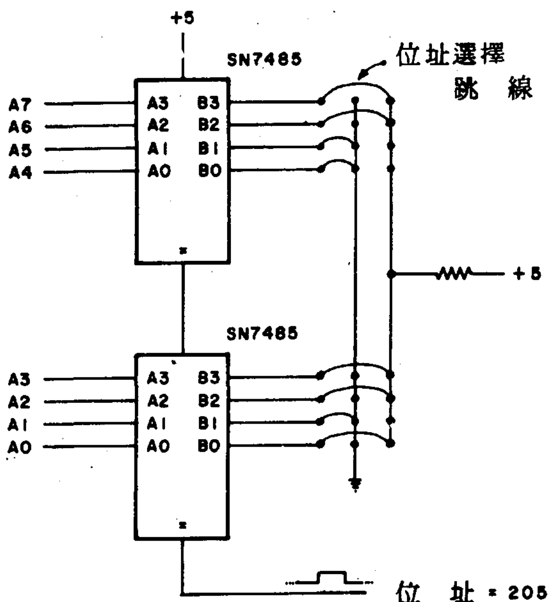


圖2－18 以兩個SN7485比較器測知位址205。

，這個圖形僅畫出8個位址位元。圖中之解碼器已接線成能測知位址205（＝11001101_2）的情形。就像一八輸入之閘控電路一樣，這個方法僅能測知單獨一個位址。因此，爲了具有更大的彈性，絕大多數情況下，比較器均如圖2－19所示般地，與解碼器合用。SN7485比較器唯一的“相等”輸出爲邏輯1，故必要時，其必須先經反相，才能作爲解碼器的致能信號。就這個圖中所示的電路而言，我們另外又加了兩個比較器，以期設備位址能絕對解碼。如此，SN74154解碼器的輸出唯有在A15～A4等位址位元與三個比較器電路之輸入處所事先設定的相對邏輯狀態完全符合時，才會呈

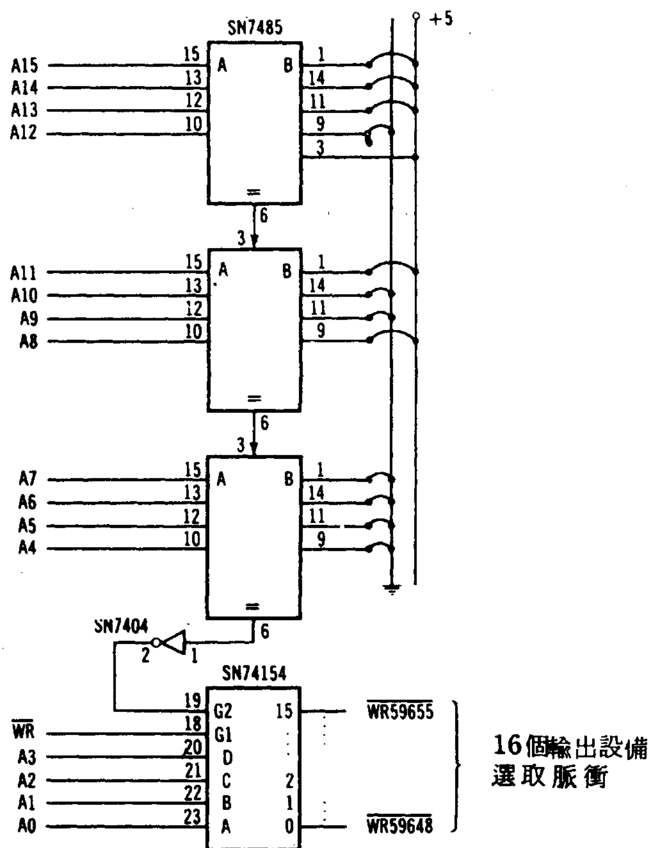


圖 2－19 以比較器與解碼器作位址選取

動作狀態。在這種狀況下，A15～A8 等位址位元必須爲 11101001 ，且 A 7～A 4 等位址位元必須爲 0000 。由於 $\overline{WR}$ 功能脈衝亦須出現，解碼器才能致能（動作），因此，您應了解，這個電路所產生的，即爲位址 59648 （=E900H）至 59655 （= E907H ）的輸出設備位址選取信號。若欲

產生 16 個設備位址選取信號，以便能同時選取輸入設備，則這個電路必須另外再加上一個 SN74154 解碼器。除了 $\overline{WR}$ 信號應換成 $\overline{RD}$ 信號外，兩個解碼器之輸入應並聯。

至此，有關設備選取電路以及組合設備位址與功能脈衝以獲取設備選取脈衝的討論就算完成了。後面再舉例時，希望您認得 $\overline{WR\ 54390}$ 就是將 $\overline{WR}$ 功能脈衝與位址 54390 經適當閘控所產生的一個邏輯 0 設備選取脈衝。有時，我們會畫出實際的閘控電路，但通常則不，作者假設您已知信號的來源。雖然您或許會在其它書刊雜誌上看到許多其它不同的設備選取電路，但迅即您會發現，這些電路的動作情形幾乎全無兩樣——位址信號與功能脈衝一同經過閘控，選取某一個特定設備。

某些實驗將讓您探求以設備選取脈衝控制設備。在下一章，您會學到如何以這些脈衝控制八位元資料位元組在6502資料巴士上的流動。

3

輸入/輸出界面

前一章，我們已介紹了數種選取與辨認輸入／輸出設備的方式，目前，該是實際組成輸入／輸出口的時候了。這一章，我們將探討一些使輸入／輸出設備能將一位元組之資料送給計算機，以及能接收計算機所送出之位元組資料的實際巴士界面技巧。正如設備選取電路一樣，同樣有許多電路可構成輸入口與輸出口。不過，我們將僅舉少數幾個例子，說明界面的基本原理。

3－1 輸出口

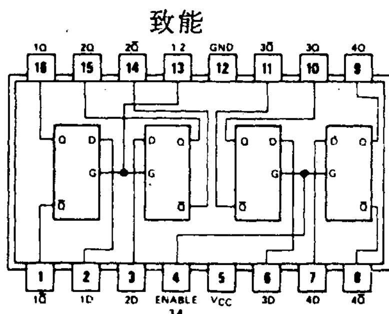
所謂輸出口即為在BASIC語言程式之POKE命令的控制下，接收計算機所送出之資料位元組的設備或元件。您已經明瞭，在POKE指令執行時，巴士上所傳送之資料──$\overline{WR}$脈衝與設備位址──間有一定的時序關係。這在圖1-4時已說過。在Apple計算機上，$\overline{WR}$脈衝的期間大約為500ns。因此，若我們以$\overline{WR}$脈衝控制將資料由資料巴士傳給輸出

設備，透過使用設備選取脈衝，則資料呈現給輸出設備的時間將僅約500ns左右。在這麼短的期間內，接收（亦即，輸出）設備誠難有所作爲（亦即，安穩地讀取資料）。爲了解決這個問題，每一輸出口都必須備有可自巴士獲取資料，並將資料"保存"至被輸出設備拿走或被另一次資料傳輸"更新"（改變之意）爲止的某種電路。

可達成此一功能的電路即稱爲鎖住器（latch）。鎖住器會栓住資訊，將之保存至被更新或電源停掉爲止。鎖住器IC有許多種不同型式，每一種型式均具有不同的控制結構以及資料輸入輸出。本章，我們將介紹其中三種最一般性的：SN7475，SN74175，與SN74LS373，而不每一種都介紹。圖3-1所示即爲這三種鎖住器之接脚圖與功能表。注意，雖然SN7475與SN74LS373都爲眞正的鎖住器元件，但SN74175實際上則含正反器（flip-flop）。由於SN7475鎖住器晶片內含四個鎖住器電路，且SN74175內含四個正反器電路，因此，組成一個八位元輸出口需有兩個SN7475或兩個SN74175 IC。但SN74LS373由於每一個晶片就含八個鎖住器電路，因此，構成一個八位元輸出口僅需一個晶片就夠了。

現在，我們先簡單地介紹一下這幾個鎖住器電路的動作原理，好讓您能毫無問題的使用。就以SN7475作例子。您可將SN7475鎖住器想成是"記憶閘"。這點由圖3-1之SN7475的功能表中即可看出。仔細看這個功能表，您會發現，當致能輸入(G)爲邏輯1時，"D"輸入所呈現的資料（或邏輯準位）直接通過鎖住器到達"Q"輸出。$\overline{Q}$輸出則爲Q

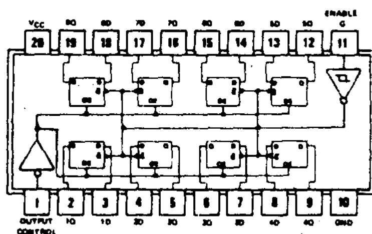
致能



功能表
（每一鎖住器）

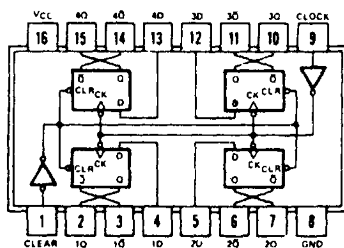
輸入		輸出	
D	G	Q	Q̄
L	H	L	H
H	H	H	L
X	L	Q_0	$\bar{Q}_0$

H＝高準位　L＝低準位　X＝無關
Q_0＝G由高電位轉至低電位前之Q準位



'LS373, 'S373
功能表

輸出控制	致能 G	D	輸出
L	H	H	H
L	H	L	L
L	L	X	Q_0
H	X	X	Z



功能表
（每一個正反器）

輸入			輸出	
清除	時序	D	Q	Q̄†
L	X	X	L	H
H	↑	H	H	L
H	↑	L	L	H
H	L	X	Q_0	$\bar{Q}_0$

圖 3－1　SN7475(上)，SN74LS373(中)，與SN74175(下)鎖住器晶片之接脚圖與功能表

輸出之反態。而當致能輸入由邏輯 1 變至邏輯 0 時，此時 D 輸入所呈現的準位則爲 Q 與 $\overline{Q}$ 輸出所記住或鎖住。圖 3-2 所

示的時序圖正好說明了這些動作。

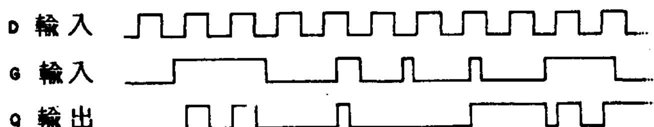


圖 3－2　SN7475鎖住器電路的時序圖

每當〝G〞輸入變爲邏輯 1 準位，Q輸出即變成〝D〞輸入之狀態，不論〝D〞輸入之準位是否正在改變。換言之，當〝G〞輸入爲邏輯 1 時，〝D〞輸入之邏輯準位傳遞至〝Q〞輸出；而當〝G〞輸入變爲邏輯 0 時，〝Q〞輸出仍然維持於原來〝D〞輸入的準位不變。這就是 SN7475 鎖住器電路的動作原理。SN7475 分成兩部份，每一部份均個別獨立動作，彼此互不相干。四個鎖住器電路欲串聯一起動作時，晶片上的兩個閘極輸入可接在一起。當然，幾個鎖住器的輸入與輸出彼此還是互不相干，如此，電路內就有四個不同來源的輸入信號。不過，若彼此欲成縱列動作，則全部四個輸入都必須在同一時間鎖住。

SN74LS373的動作原理與SN7475類同，但其却僅具有一個閘控或致能信號，此外，這個晶片亦僅有Q輸出，而無 $\overline{Q}$ 輸出。SN74LS373 另外具有一個輸出控制，不過，當 SN74LS373 被用作輸出口時，這個控制信號（第 1 支接脚）通常接地。

SN74175 晶片則含有四個正反器，這四個正反器獲取

並持住於時序脈衝正向緣（positive-going edge）時出現於輸入端的資訊（邏輯準位）。四個正反器的輸出唯有在這個時候——亦即，時序脈衝正向緣時——才會改變。其它時間，不論時序信號爲邏輯0或邏輯1準位，SN74175 的輸入均不致影響輸出。此即此一正反器元件與其它之鎖住器元件的不同所在。雖然如此，但兩種晶片在計算機界面上的功用則無異。

爲了能〝清除〞正反器的狀態（Q＝0，$\overline{Q}$＝1），SN74175 晶片上通常有一個共用的清除輸入（第1支接脚）。每當這個輸入加邏輯0準位時，每一個正反器即呈清除狀態（Q＝0，$\overline{Q}$＝1）。平常，這個清除輸入均接＋5伏特（邏輯1）不用。

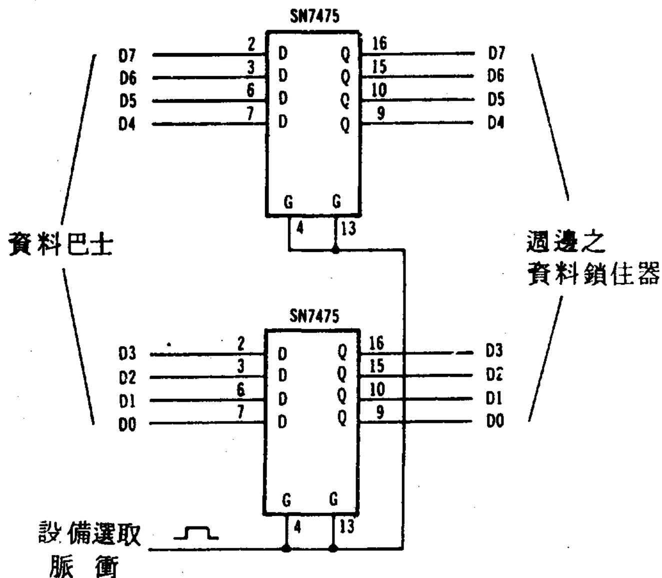


圖3－3　以兩個SN7475鎖住器晶片組成一個輸出口

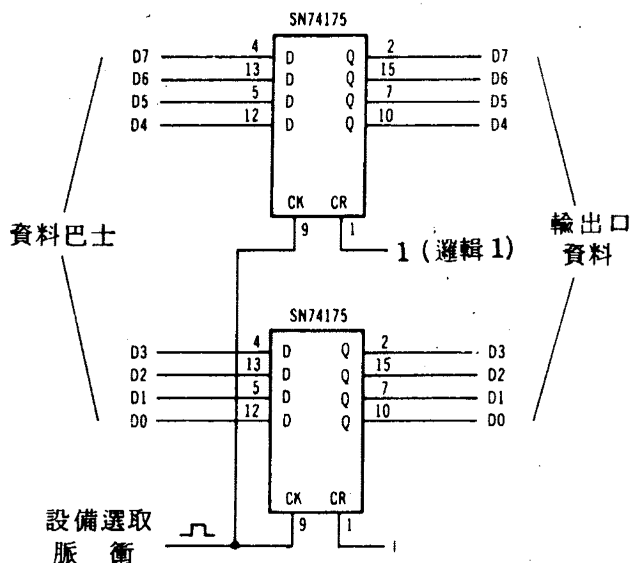


圖 3－4 以兩個SN74175鎖住器晶片組成一個輸出口

在執行 POKE 命令時，這三個積體電路之任一者均可用以鎖住Apple計算機所輸出之資料。一經正確接在巴士上後，以一輸出設備選取脈衝推動此一鎖住器電路就不是什麼困難的事了。圖 3-3 所示即為一典型的八位元輸出口。在這個電路內，欲教鎖住器電路捕獲且持住Apple所輸出的資訊時，您必須加上一準位為邏輯 1 的輸出設備選取脈衝。

圖 3-4 所示則為以兩個 SN74175 鎖住器晶片構成輸出口的情形，這個電路以某種邏輯監視器顯示出晶片所鎖住的資訊。輸出口清除輸入接脚處的" 1 "代表這些輸入均連至＋5 伏特，或邏輯 1 準位。我們特別以這個" 1 "代表邏輯準位連接，以有別於電源供給連接——接至電源供給時通常寫成＋5 伏特或＋5 V 。

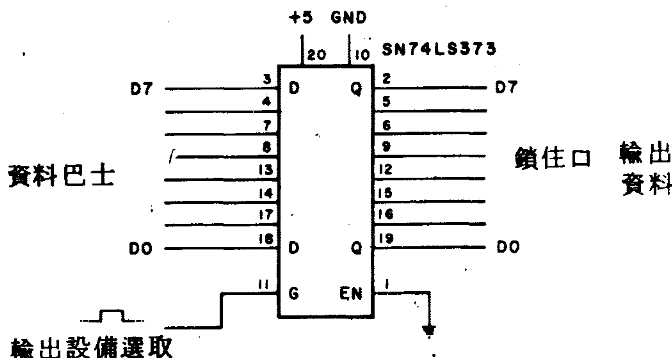


圖 3－5　以SN74LS373鎖住器晶片構成一輸出口

圖 3-5 所示則爲以一SN74LS373鎖住器晶片（內含八個鎖住器電路）構成一個八位元輸出口的情形。使用這種IC時，只要一個晶片即可構成一個輸出口。爲了使輸出永遠保持致能，輸出控制線（EN）特別接地。同樣地，設備選取電路必須供應一個邏輯 1 準位的輸出設備選取脈衝。這個輸出口一旦正確連接至資料巴士，而且設備選取脈衝亦有來源後，其即可在軟體命令的控制下開始使用。舉個例子而言，POKE　49312, 0 命令執行的結果，計算機將一爲零的數值送至位址 49312的輸出口。若事實上有一個輸出口連接至資料巴士，而且其位址爲 49312，則這個零就會出現在這個輸出口上。

例題 3－1　49320輸出口上之八位元二進計數程式

```
10  FOR N = 0 TO 255
20  POKE 49320,N
30  NEXT N
40  GOTO 10
```

例題 3-1 的程式即可用以在位址 49320的輸出口上產生一逐次累增的二進計數。整個計數將循 255,0,1,2, ………254,255,0,1 ……等等的順序，綿延不斷。稍後在作實驗時，您會再度看到這個程式。

由以上的敍述讀者可看出，輸出口極易構成。絕大多數並行輸入且並行輸出的邏輯元件，只要具有內部鎖住能力，都可用作鎖住器。舉個例子而言，SN74193 可規劃二進計數器，SN74LS194 A萬用移位暫存器，SN74198 移位暫存器等等，就都可用作鎖住器。

雖然絕大多數的輸出口均能輕易地以標準規格的積體電路構成，但最近市面上也出現了一些指明專供微電腦使用的新型積體電路，這些電路內部均具有鎖住的功能。內含一個鎖住器之 Signetics NE5018八位元數位至類比轉換器晶片就是其中一個例子。

輸出口的典型應用包括下列幾些：

送出資料給印字機

送出資料給顯示幕

控制紅綠燈

送出資料給軟性磁碟

推動模型鐵道之開關

控制一化學程序中之閘門與唧筒

控制繪圖機

送出資料給七段顯示器

控制另一部計算機

有些應用實際使用資訊的數值，有些則僅個別使用每一位元之狀態（開或關，即 1 或 0 ）。但像印字機之類的設備

則可能兩種都用：既有傳送欲印出資料的口，亦有控制印字機之動作的口。七段 LED 所構成的顯示器，雖然僅算一種〝設備〞，但其通常必須使用數個輸出口。

3－2　輸入口

輸入／輸出設備使用輸入口，以便將資訊傳給計算機。不像輸出口必須能接收並持住計算機於某一時刻置於資料巴士上的資訊，並且必須隨時與資料巴士接在一起，輸入口在不用時，必須能與巴士〝斷接〞。雖然輸入口必須將邏輯 0 與邏輯 1 送給 CPU，但其卻必須配置成當其不被選取時，不致於干擾巴士之使用的情況。

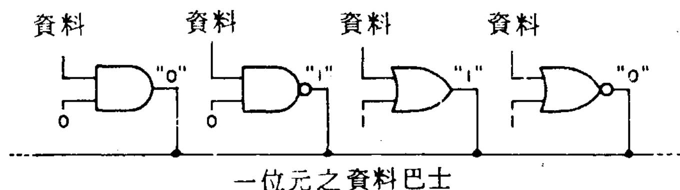


圖 3－6　試著將標準邏輯閘直接連在巴士上

如圖 3-6 所示，由於〝未選取〞之輸出狀態可能爲邏輯 0，亦可能爲邏輯 1，因此，我們無法直接以簡單的邏輯閘將資料〝置於〞資料巴士上。注意到，即使沒有任一個閘被選取或致能，正如以引號括起之邏輯準位所示的，每一邏輯閘之輸出準位還是各不相同。這些準位彼此會〝爭著〞使用巴士，因而很可能導致燒壞某一個或某幾個晶片。此即清楚

地說明了為何邏輯閘不能直接用在資料巴士上的原因。

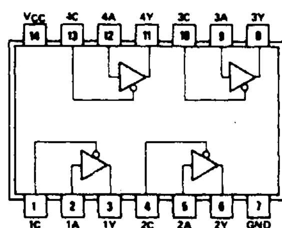


圖 3 — 7 SN74125巴士緩衝器晶片之接腳圖

具有三態(three-state or tri-state)輸出的特殊積體電路則可解決此一問題。典型的三態元件為如圖 3-7所示的 SN74125 巴士緩衝器。這個晶片內含四個完全一模樣的緩衝器。每一個緩衝器(輸出準位等於輸入準位)均具有一額外控制線,在符號上正好連接至三角形符號的一邊。緩衝器的動作情形是,在致能時,輸入端之邏輯準位原樣傳遞至輸出端。不過,與一般簡單邏輯閘不同的,當被禁能(disabled,亦即,不動作)時,緩衝器之輸出端與巴士(或其所連接的設備)形成"斷接"。這就是三態元件的所謂第三態(另兩種狀態分別為邏輯0與邏輯1),這種狀態經常稱為高阻抗狀態(HI-Z)。三態元件的連接與斷接速度相當快,一般均在20ns以內。

在SN74125 內,每一個緩衝器均有其各自的致能輸入。在資料欲由輸入傳遞至輸出時,緩衝器的致能輸入必須加邏輯0。若致能輸入加邏輯1,緩衝器之輸出即呈高阻抗(或斷接)狀態。另一個類似的電路SN74126,接腳則與SN74125 完全相同。不過,其致禁能控制則正好相反——

致能輸入加邏輯 1 時，緩衝器致能，加邏輯 0 時緩衝器禁能（呈高阻抗狀態）。我們舉這幾個晶片主要在說明三態元件的動作原理。這些晶片在實際計算機界面電路上很少見，因爲，還有其它更好用的元件存在。

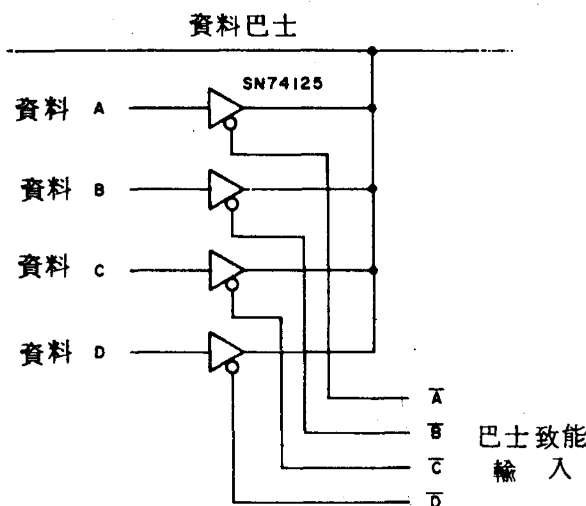


圖 3－8　典型三態巴士，四條線

表 3－1　四線之三態巴士的眞值表

致能 D	C	B	A	巴士內含
1	1	1	1	無法決定（所有設備均 HI－Z）
1	1	1	0	資料 A
1	1	0	1	資料 B
1	0	1	1	資料 C
0	1	1	1	資料 D
0	0	0	0	不允許

我們舉一個典型的巴士來作說明。於圖 3-8 之電路內，巴士上接了四個一位元的設備。雖然八位元的系統總共需有八條線，但為了清楚起見，我們僅畫出了單位元的巴士。當四個巴士致能輸入中有任一者加邏輯 0 準位時，其所對應的資料輸入即會通過緩衝器，到達巴士上。我們假設沒有其它設備連接在巴士上，因此，表 3-1所示的眞值表正好適用於這個簡單的巴士電路。

倘若四個緩衝器均未被致能或接至巴士，則除了邏輯閘輸入、記憶器等資料位元的〝接收者〞外，巴士並未連接至任何東西上，因此，巴士的邏輯值未知。而只要巴士緩衝器之致能輸入有任一者加邏輯 0 ，這個被選取的緩衝器即會將其資料送至巴士上。注意，不能同時有一個以上的緩衝器被致能，因為，這樣會造成巴士衝突。

Apple計算機系統中，所有用以將資訊傳給 CPU 的設備均必須具有三態輸出。是以，即令記憶器亦須具有三態輸出（事實上亦然）。此時，計算機的設計者就必須負責保證，系統中不會同時有兩個或兩個以上的輸入設備被選取。否則，若發生這種複選的情形，計算機即無法正常的作業。

由此可見，用以將資訊傳給計算機的輸入口可輕易地以標準的三態IC構成。大部份情況下，我們都使用八個個別的三態緩衝器，每一條線一個。同樣地，大多數情況下，致能輸入亦都並行連接，以使八個緩衝器能將資訊同時送出至巴士上。這個共用的致能輸入有時會做在晶片內，以期僅以一支接脚控制到全部八個位元。

雖然有許多晶片都可用以組成輸入口，但其中僅有少數能一般化至符合我們的考慮。我們將使用的兩個主要積體電

路是 SN74365 與 SN74LS244。SN74365亦等於DM8095（國際半導體公司產品）。這兩個晶片的接脚圖如圖 3-9 所示。

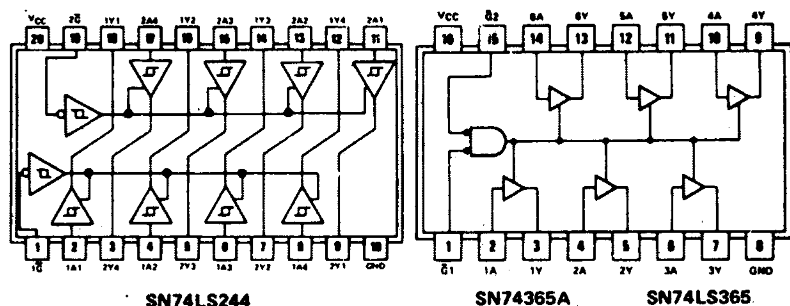


圖 3－9　SN74LS244與SN74365(DM8095)
三態巴士推動器晶片的接脚圖

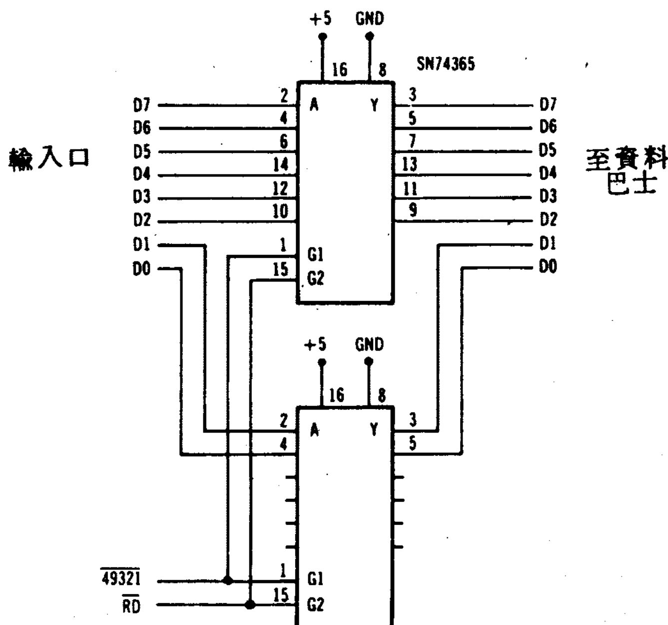


圖 3－10　以SN74365晶片構成的典型輸入口

迅速您會發現，SN 74 LS244 晶片上含有八個三態緩衝器，而SN74365 晶片上則僅有六個。因此，若我們以SN74365 晶片構成輸入口，我們就必須使用兩個 IC。圖3-10 所示即為一典型的八位元輸入口。就這個構造而言，下面的 SN74365 晶片則僅用了兩個三態緩衝器。由於SN74365 內含有控制三態緩衝器之致能的NOR閘，因此，我們特以之閘控 $\overline{RD}$ 功能脈衝與設備位址 $\overline{49321}$。倘若設備選取信號 $\overline{RD\ 49321}$ 已在界面電路的其它地方產生，則您可將這個信號加至兩個晶片上之其中一個致能輸入，然後將另一個致能輸入接地，或接邏輯 0。這種控制方法即如圖3-11所示。

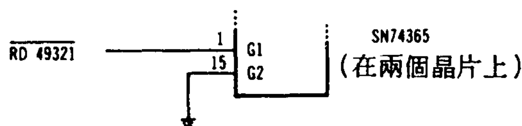


圖 3－11　SN74365三態晶片的另一種控制方法

如例題 3-2 所示，使用這樣一個輸入口，我們可以PEEK命令將資料值輸入至計算機。

例題 3－2　49321口之資料輸入程式

```
10  A = PEEK (49321)
20  PRINT A
30  GOTO 10
```

就這個例子而言，當 10 號一列之 PEEK 命令輸入一個

八位元的二進值時，這個數值即被轉換成一介於 0 至 255 之間的十進數，然後〝印出〞在顯示幕上。其功用與下面述句所達成者完全相同：

```
10 PRINT PEEK(49321): GOTO 10
```

使用具有八個（八位元）緩衝器之 SN74LS244 晶片亦可組成一類似的輸入口。這個晶片包含兩組互相獨立的緩衝器，每一組各有四個緩衝器。這兩組緩衝器又分別由兩個致能輸入，$\overline{2G}$ 與 $\overline{1G}$ 所控制。由於 SN74LS244 無內附的NOR閘，故設備選取閘控必須設在外面。圖 3-12 所示即爲一以

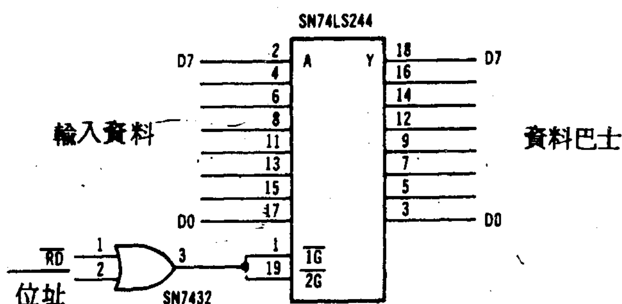


圖 3－12　以SN74LS244構成的輸入口

SN74LS244 晶片所構成的典型輸入口。例題 3-2 所示的程式同樣可用於控制將資訊由這個輸入口輸入至計算機。

SN74365 與 SN74LS244 兩者亦皆有接脚完全一一對應，但資料位元通過晶片時却被反相再置於資料巴士上的類

似晶片。這兩個晶片分別是SN74366與SN74LS240。此外，SN74366亦對等於DN8096晶片。一般而言，界面電路中所使用的緩衝器均為不反相緩衝器。

偶而，週邊設備所產生之資訊會超過8位元，而這些位元必須都為計算機所讀取。像12位元的類比至數位轉換器就是其中一個例子。在這種輸入資訊超過八位元的情況下，我們必須將輸入資訊以八位元為單位分組。就前述之12位元轉換器而言，輸入資訊必須分成兩組，一組含八位元，另一組含四位元。同樣地，十六位元值亦需使用兩個輸入口，九位元值亦然。在輸入口之八個位元並非全部使用的情況下，未用的位元通常都接地，使其變為邏輯0狀態。倘若這些未用位元的狀態無法決定，則很可能這些位元根本就不做在輸入口電路內。只要適當地使用軟體命令，您同樣可"除去"這些位元。這時，您只要以命令"遮掉"（mask）這些未用位元，使其全變為0就可以了。

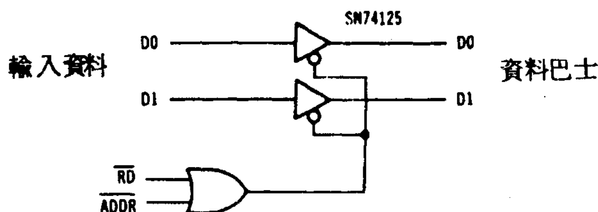


圖3－13　兩個位元之輸入口

至此，輸入資訊超過八位元的輸入口結構都已定義過了。接著，讓我們來看看如何將自兩個輸入口（圖3-13）所

輸入的兩個資料位元組重新組合，還原成原來的數值。最低次的八位元由於Apple計算機在輸入的過程中即已自動將之轉換成一值介於 0 至 255 之間的十進數，因此，這八個位元沒問題（其值本來就等於一介於 0 至 255 之間的十進數）。不過，較高次的四位元（就12位元轉換器的例子而言）就有問題了。這四個位元若單獨考慮，則其將被轉換成一介於 0 至15 間的十進數，而不是其原有的位置值 256，512，等等。不過，很明顯的，由於這四個位元是一個十二位元之資料值的高次四位元，因此，其值剛好差一個 256的因子。記得，Apple所輸入的任何八位元值，均被自動轉換成一介於 0 至 255 之間的十進數。

根據以上的討論，Apple所輸入的兩個數值就很容易能重組成原來的資料。我們只要將高次四位元的值乘以 256，然後再加上低次八位元轉換所得的值，即可獲得原來界面設備所產生的十二位元資料值——爲一介於 0 至 4095 間的數值。達成此一作業的完整軟體常式即如例題 3-3 所示。

例題 3－3　12位元輸入的轉換程式

```
10  A = PEEK(49312)
20  B = PEEK(49313)
30  C = (B * 256) + A
40  PRINT C
```

若將所有步驟都寫在一列上，則這個程式就變成下面的單獨一列：

```
10  PRINT(PEEK(49313) * 256) + PEEK(49312)
```

執行時，這個程式將印出程式執行時，呈現於週邊或界面設備之12位元二進值的對等十進數。這個程式適用於具有 9 至 16個二進輸出的任何界面，不過，未用的位元必須仔細地接地。在後面實驗中，您會看到另外採用"罩蓋"(masking)的方法。

輸入口的用途是將外部設備的資訊傳給計算機。這個資訊可能代表一個重量、一種溫度、或一項電阻值。當然，其亦可解釋爲許多分別代表設備之狀態(開或關)的二進位元。以下所列即爲輸入口的一些典型用途：

將交通察覺訊息送給計算機
將某一儀器所產生的數位值送給計算機
將印字機的狀態(開/關)位元送給計算機

在界面應用上，輸入口的主要要求是，其輸出必須爲三態控制，以期不致在資料巴士上造成衝突。

4

旗號與決策

幾乎在前面的所有例子內，我們都假設計算機與外部輸入／輸出設備間幾乎不需有任何同步。因此，輸出口永遠可立即接受計算機所送來的資料，且當計算機執行至程式的 PEEK 命令時，輸入口隨時都有正確的資料可讀取。事實並不盡然。經常，我們都必須處理輸入／輸出設備比計算機速度慢的情況。

4－1　輸入/輸出設備的同步

由於並非所有的輸入／輸出設備隨時都可立即應付計算機的作業（輸入或輸出），因此，計算機與輸入／輸出設備間必須有某種彼此取得同步（協調）的方式。這種同步通常涉及使用稱爲旗號（ flag ）的信號。這些信號用於顯示每一設備的狀態——忙碌或空閒，準備好或未準備好等等。換言之，這些信號即代表了設備的狀態（此乃爲何旗號經常被稱爲狀態旗號的理由）。

爲了舉例說明方便起見，我們假設我們欲將一個設備界面（連接之意）至Apple計算機上。這個設備不定期地供應八位元的資料值給計算機。通常，在準備將資料送給計算機時，這個設備同時亦會產生一個旗幟信號。圖 4 － 1 所示即爲這個設備。注意，其以一標準的三態輸入口將資訊送給計算機。READY（已準備好）旗號產生了一個很有趣的問題。那就是，計算機如何監聽或查看READY 旗號的狀況，以得知又有新的資料值欲輸入呢？

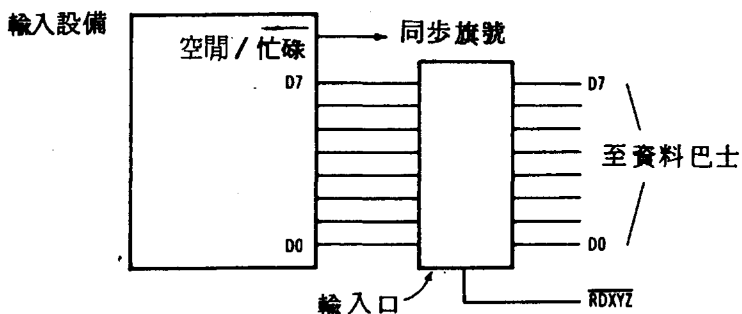


圖 4 － 1　附有同步旗號輸出的簡單輸入設備

前面我們曾經說過，輸入口並無限制說不能傳遞眞正的數值。事實上，計算機根本不知道它所正傳遞的八位元值，譬如 01100100_2 ，是代表數目 100，或是一項代表五個設備關三個設備開的狀態訊息。因此，我們可設置另一個輸入口，將輸入設備的狀態訊息傳給計算機。這個輸入口僅用一個位元就可以了，其它七個位元可以不用，或用以顯示其它外部設備的狀態。如此，我們即可以軟體步驟來查驗外部設

備的狀態了。

在以計算機程式檢查旗號狀態時，我們可將程式設計成，教計算機一直等到旗號變成某種狀態時，再進行某種必要的作業；或者，程式亦可設計成週期性地檢查旗號狀態，讓計算機能同時做其它的計算。

不論 BASIC 語言或組合語言，兩者均有能檢查個別旗號狀態或一八位元字組之個別位元的邏輯運算指令。使用這些指令，我們可輕易地測知某一旗號究竟是呈邏輯 1 或邏輯 0，以讓計算機根據旗號的狀態作決策，採取適當的進一步行動。

4－2　邏輯運算與旗號

就測知旗號狀態而言，最有用的運算或許就是 AND 運算了。您應還記得，兩個位元 A 與 B 可以“ AND ”在一起。如圖 4－2 所示，這兩個位元 AND 運算的結果，唯有當

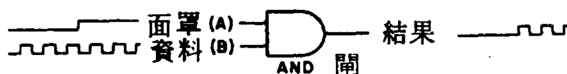


圖 4－2　邏輯 AND 運算：以 DATA(資料)與 MASK(面罩)產生 RESULT(結果)。

兩個位元同時為邏輯 1 時，輸出結果才為邏輯 1，否則，其它情況輸出結果均為邏輯 0。另一種看法是，將“ A ”輸入

視爲“罩蓋”(mask)位元，而將“B”輸入視爲“資料”輸入。當我們欲遮蓋掉資料位元時，我們只要令面罩位元爲0，輸出結果就一定爲0。只要面罩位元爲1，資料位元就原封不動呈現於輸出端。如此一來，我們即可隨意地將某些位元遮蓋掉，而讓某些位元順利“通過”。舉個例子而言，假設資料字組 00111010 中之第五位元(D_5)代表我們所欲檢查之某一設備的狀態，則我們就可使用 00100000 的面罩，讓這兩者 AND ，以單獨分出我們所要的位元。這時，經 AND 運算後，倘若結果爲零，我們即可知道我們所要測知的狀態位元值亦爲邏輯0；否則，其即爲邏輯1。這個測試結果就可作爲程式進一步作決策的根據。

特別記得，在設計 BASIC 程式時，面罩一定要轉換成十進等效。就剛的例子而言，面罩就是十進數 32 。

4－3　測知旗號的軟體

界面一旦像圖 4－4 般地做好，讓各種旗號能被測試時，軟體即可根據這些旗號的狀態作決策。

旗號值	00111010	00011010	11110000	00011111
面　罩	00100000	00100000	00100000	00100000
結　果	00100000	00000000	00100000	00000000

圖 4－3　兩個八位元字組，位元對位元作 AND運算

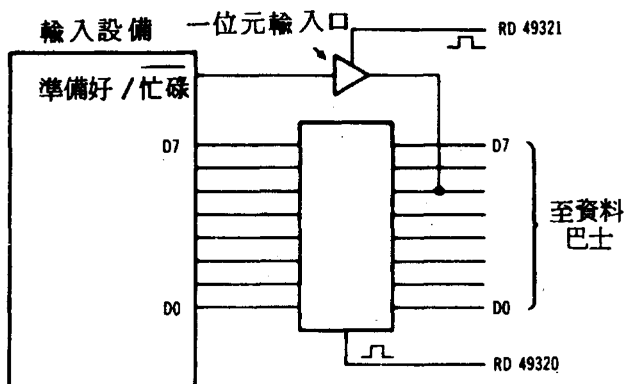


圖 4 － 4　旗號可以軟體測知的完整界面

有些 BASIC 方言具有能作位元對位元 AND 運算（如圖 4 － 3 所示）的邏輯運算指令。在這些情況下，BASIC 程式即可以很簡單的式子，對兩個值均介於 0 至 255 之間的資料字組作邏輯 AND 運算。記得，雖然程式上寫的是十進數，但眞正被“ AND ”的是其二進等效。例題 4 － 1 與 4 － 2 所示即爲以 AND 運算測知自 49321（位址）輸入口之 D_5 位元所輸入的旗號狀態情形。

例題 4 － 1　以邏輯 0 旗號作控制

```
4010  A = PEEK(49321)
4020  IF (A AND 32) = 0 THEN 200
4030  ··· 若旗號爲邏輯 1 ，則繼續執行這兒的指令
```

例題 4 － 2　以邏輯 1 旗號作控制

```
4010  A = PEEK(49321)
4020  IF (A AND 32) > 0 THEN 200
4030  若旗號爲邏輯 0 ，則繼續執行這一列上的指令
```

不論那一種情況，每當適當的條件滿足時，程式即跳去做某種此時應該做的工作，譬如，自輸入口輸入資料等等。

可惜，Apple計算機的邏輯運算並不這樣用。在Apple，AND 運算只能對兩個不同的眞假狀況作 AND ，因此，其就很難遮掉一八位元字組之某七個位元，而求知某一位元的狀態。除非您願意花費很多時間，否則，您會發現以組合語言副程式來做這個運算相當方便迅速。由於組合語言常式用來極爲簡便，因此，我們想順便在這裡提一下。事實上，後面我們將提供您許多簡單且易於使用的常式。

4－4　組合語言邏輯運算

6502 微處理器的指令集中含有一個 AND 運算與一個 OR 運算。這兩個指令均對兩個八位元的資料運算，並產生一八位元的結果。因此，我們必須寫一個做這個運算的簡短常式。

由於Apple在第 03H 個記憶頁上有一些“空閒”可讀寫記憶位置，因此，我們選擇將這些常式置於這一頁上。如此，常式即可與您所使用之計算機的記憶器大小無關。這個常式的全部列表即如表 4－1 所示。注意，表中特別列出了十六進與十進位址與資料／指令值。您即使不懂組合語言程式設計，同樣亦可使用這個常式。不過，爲了讓您知道這個程式如何發生作用，我們特別作了一些說明。

常式使用三個可讀寫記憶位置暫時儲存 MASK

表 4－1　組合語言邏輯副程式

位址位元組		資料位元組		
十六進	十進	十六進	十進	
0300	768	—	—	面罩位元組置於此。
0301	769	—	—	資料位元組置於此。
0302	770	—	—	答案置於此。
0303	771	48	72	PHA A暫存器內含存起
0304	772	AD	173	LDA 自面罩位置取入A
0305	773	00	0	
0306	774	03	3	
0307	775	2D	45	AND Reg A 與 DATA *
0308	776	01	1	
0309	777	03	3	
030A	778	8D	141	STA 結果存入答案位置。
030B	779	02	2	
030C	780	03	3	
030D	781	68	104	PLA 取回A之內含
030E	782	60	96	RTS 回至 BASIC

* OR運算時代換成ODH或十進數13。

BYTE、與ANSWER等三個資料位元組。MASK 位置儲存面罩位元組， BYTE 位置儲存欲運算的位元組。邏輯運算後，所得結果則儲存於ANSWER的位置上。

使用這個常式前，您必須將面罩位元組存入位址768的記憶位置，且將欲運算的資料位元組存入位址769的記憶位置。這可以 POKE 命令達成。這些資料存好後，您只要叫用（ call）這個組合語言副程式，所需運算即會自動完成。至於，如何叫用這個副程式呢？

從BASIC叫用一個組合語言副程式並不難。在Apple計算機內，您只需將一個三位元組的跳越（ jump ）指令存入位址10 ，11 ，與12（表示成十六進制則為OA ，OB ，與OC ）三個連續記憶位置就可以了。由於副程式儲存於位址771（＝0303H）起之記憶位置，因此，這三個位置所

必須儲存的資訊分別為：位址 10 存 76 ，位址 11 存3，位址 12 亦存3。這三個位置的資訊一旦存好了，您即可以 USR 函數叫用這個組合語言副程式。當然，您必須先將面罩與欲運算之資料位元組存入剛剛提過的位置，然後才能使用 USR 函數。這個情形即如例題 4 － 3 所示。

例題 4 － 3　叫用邏輯運算副程式

```
1590  POKE 768,32: POKE 769,129
1594  Q = USR(5)
```

就例題4 － 3之例子而言， 32 為面罩位元組，而 129 則為欲與 32 作 AND 的資料位元組。Q乃一使用 USR 函數所必須的“虛擬”（ dummy ）變數，且 5 亦為一毫不影響副程式的“虛擬”值。只要在其它地方不用，則 Q 可代換成任何一個變數，同樣地， 5 亦可換成任何其它數值，譬如說 0 。

叫用過組合語言副程式後，邏輯 AND 運算的結果就儲存於位址 770 的記憶位置內。這時，您即可以 PEEK 命令獲得這項結果。例題 4 － 4 所示即為這個副程式的完整用法。當然，我們假設組合語言副程式已事先儲存在記憶器內，譬如使用監督程式。程式的第一列述句將三位元組的跳越指令存入位址 10 ， 11 ，與 12 三個連續記憶位置內。然後，第二列述句再將面罩位元組與資料位元組分別存入適當位置。您可看出，欲運算的資料位元組以 PEEK 命令由一輸入口獲得。

例題 4－4　邏輯運算副程式的完整用法

```
2030  POKE 10,76: POKE 11,3: POKE•12,3
2040  POKE 768,32: POKE 769,PEEK(49321)
2050  Q = USR(7)
2060  IF PEEK(770) > 0 THEN 3460
2070  ···若旗號爲 0，則繼續執行此一列指令
```

只要將AND運算的運算碼（ op-code ）2DH改成OR運算的運算碼 ODH ，剛剛的副程式即可做邏輯 OR 運算。同樣地，這亦可在副程式使用前以一 POKE 命令達成。由此可見，表 4－1 所示的副程式既可做 AND 運算，亦可做 OR 運算。

讀者您應自己會使用Apple計算機的監督程式（ monitor ），將我們所要的副程式取入可讀寫記憶器內（第 03H 頁）。監督程式的用法請參閱參考手冊。事實上，您亦可以 12 個 POKE命令將這個副程式存入記憶器內，不過，這種方法極易造成錯誤。因此，勸讀者還是少用。

很可惜，由於Apple之BASIC無法直接使用這些邏輯運算，因此，您得必須假手於組合語言副程式。不過，所幸這個組合語言副程式倒還十分容易使用。其用法就是以上我們所舉例說明的。

4－5　複雜旗號

至此，您或許會問，倘若圖 4－4 之輸入設備上的旗號用以顯示輸入設備有一八位元資料欲送出，那輸入設備又怎

能知計算機是否已讀取其所欲送出的資料呢？這個問題的答案是這樣的。計算機在讀取資料後，可送一個信號給輸入設備，告訴它，資料已被讀取。而我們一般的作法都以這個信號“清除”掉旗號。這個旗號清除動作可另以一個信號達成，亦可以輸入口之控制信號一併達成。圖4－5所示即爲這種狀況，而圖4－6所示則爲一簡單的時序圖。

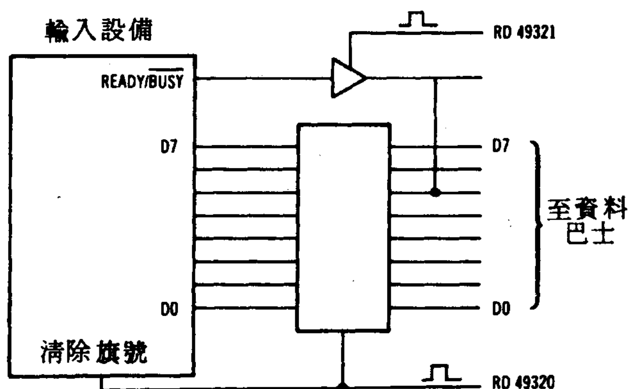


圖4－5　以計算機產生之脈衝清除旗號的完整旗號電路

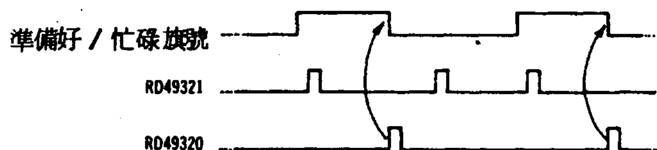


圖4－6　旗號之時序圖

當旗號爲邏輯1時，即表示設備有資料要送給計算機。

RD 49321 脈衝代表旗號狀態資訊傳遞至計算機。當計算機測試旗號且發現其值爲邏輯1時，其即會執行程式步驟，將資料自設備讀入計算機。這時候，RD 49320 脈衝的功用就是在正確的時間使三態緩衝器致能（或動作）。同時，這個脈衝亦負責清除設備內部的旗號。

第二個RD 49321 脈衝（見圖4－6）再度讀取旗號狀態，不過，由於這時旗號值爲邏輯0（表設備無資料欲送出），因此，計算機未採取任何進一步行動。不過，當第三個RD 49321 脈衝又出現時，由於旗號爲邏輯1，因此，資料再度由設備傳至計算機，而且旗號被清除。例題4－5所示即爲一可用以控制這個界面的簡單程式。假設邏輯 AND 副程式以及三個位元組的指示器（ pointer ）均已存入。

例題 4－5　簡易的旗號測試程式

```
1050  POKE 768,32: POKE 769,PEEK(49321)
1060  Q = USR(0)
1070  IF PEEK(770) = 0 THEN 50
1080  D = PEEK(49320)
1090  ... 資料輸入後繼續執行此一指令
```

以這種方式使用旗號的常見設備包括鍵盤、軟性磁碟、類比至數位轉換器、以及其它不定期地產生資料位元組的設備。

4－6 旗號電路

有時，輸入／輸出設備內部並無簡易旗號控制之必要旗號電路，否則，其所產生之邏輯準位，穩定時間就太短，無法為計算機所正確測知。在這些狀況下，“旗號”可能是一個很短暫的脈衝。事實上，若僅以三態輸入口輸入，有些旗號脈衝根本就短得無法被計算機測知。

在這種狀況下，我們就必須設計一種能“抓住”旗號脈衝，使其稍後能為計算機所測知的電路。否則，即令計算機幾個微秒就能測試一個旗號位元，其還是經常會“錯過”僅有數微秒長的短暫脈衝。

記憶旗號脈衝一般均使用正反器或鎖住器電路。典型的正反器元件如 SN7474 D型正反器，以及 SN7476 J-K 正反器。這些元件在絕大部份的數位電子學書上都有介紹。

圖 4－7 所示即為一典型的正反器旗號電路。在這個電路內，輸入設備產生之 READY（資料準備好）脈衝加至正反器之時序輸入，控制將D輸入之邏輯準位傳遞至Q輸出。計算機則經由一輸入口（該輸入口不同於傳遞資料位元組的輸入口）測知Q輸出之準位。誠如前面所說過的，計算機很輕易即可預知旗號位元的狀態。此時，一旦必要的動作都做完之後，資料即由輸入設備輸入至計算機，然後，旗號正反器被清除為 0。旗號的清除是以在正反器的清除輸入（CR）端加上一邏輯零的脈衝 $\overline{\text{CLEAR}}$ 達成。雖然控制八位元輸入

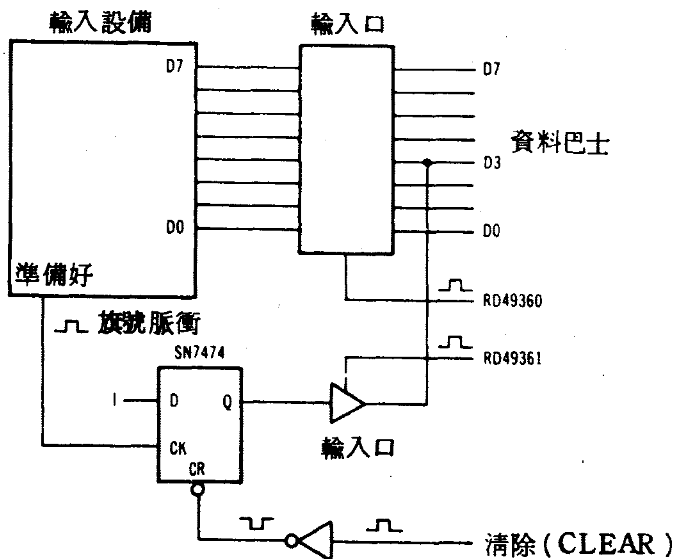


圖 4－7 用以測知旗號脈衝的正反器電路

口的 RD 49360 脈衝亦可用以清除正反器，但我們則將之個別畫成一個信號，以便能畫出如圖 4－8 所示的時序關係圖。

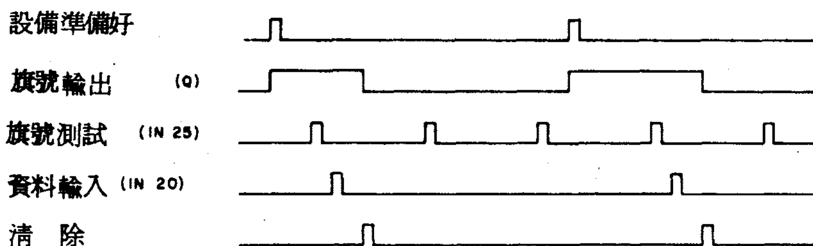


圖 4－8 旗號正反器的時序圖

在這個時序圖內，READY（備好）脈衝設定了正反器

，因此，Q輸出爲邏輯1。計算機自49361的輸入口輸入這個狀態旗號訊息，並加以測試。由於旗號狀態爲邏輯1，故計算機執行必要的程式步驟，將輸入設備上的資料輸入計算機，並且清除旗號。雖然CLEAR信號可以 POKE 命令以及適當的電路產生，但使用現成的RD 49360脈衝可能更容易一點。

就圖4－8之例子而言，旗號在邏輯0狀態時被測試兩次。由於兩次測試均顯示輸入設備無進一步新的資料，故在這一段期間內，無任何資料傳遞或旗號清除作業發生。

本書末了的許多實驗均牽涉到旗號的使用。

4－7　多個旗號

許多系統都具有數個必須不斷檢查的旗號。有時，由於某些設備較其它設備重要，必須先受注意，因此，必須建立一套優先順序（priority）。由於各旗號位元先後被測試的順序就代表了那些設備比其它設備先受注意，因此，優先順序在程式內很容易建立。圖4－6所示的程式步驟即爲一循序測試數個旗號位元的例子。這個程式依序測試D7至D5等幾個位元，因而，在這些位元所對應的設備之間，建立起一套優先順序。

例題 4－6　建立優先順序的旗號測試程式

```
300  POKE 769,PEEK(54098):POKE 768,128: Q=USR(0)
305  IF PEEK(770) > 0 THEN 1050
310  POKE 768,64:Q=USR(0)
315  IF PEEK(770) = 0 THEN 20
320  POKE 768,32:Q=USR(0)
325  IF PEEK(770) = 0 THEN 1010
330  ...其它位元同理類推
```

在這個例子裡面，D7 位元所代表的旗號爲邏輯 1 時，表示設備準備好。相反地，其它兩個旗號則爲邏輯 0 時表示設備已準備好。程式可再加入其它旗號的測試步驟，而且，旗號位元被測試的順序隨時亦可改變。留意到，有關 AND 運算的資料並未改變，而且，其只需在指令系列一開始時輸入一次即可。

4－8　插斷（Interrupts）

有時候，輸入／輸出設備在一準備好時即需立即受到服務。其根本無法等待計算機慢慢地檢查旗號，然後再根據旗號狀態作決策的幾毫秒鐘。在這種情況下，幾乎所有的計算機都至少具有一個可應付此種立即“需求”的插斷（interrupt）輸入。Apple計算機所使用之 6502 處理器晶片上具有兩支插斷輸入接脚：一支爲插斷請求輸入（IRQ），另一支爲非罩蓋（nonmaskable）插斷輸入（NMI）。其中，IRQ 輸入準位觸發（邏輯 0 狀態），而 NMI 輸入則爲邊緣觸發（edge sensitive）——受邏輯 1 至邏輯 0

邊緣觸發。基本上，這兩個輸入在Apple計算機內都不用。不過，其在內部界面接點上可以找得到，而且可於再加上週邊設備與界面時使用。

倘若某一設備需要極快速的服務——快到必須使用插斷，則我們就必須非討論不可。由於這超出本書的範圍，因此，若讀者有興趣，希望您進一步參考"**6502 程式設計與界面實驗**"以及"**6502 軟體設計**"（儒林圖書公司出版，陳金追譯）兩本書。這兩本書對插斷都有詳細的討論，並附有控制插斷的例題與組合語言程式。

Apple插斷 IRQ 與 NMI 均使用特定的記憶位置，儲存每一插斷之服務（處理）常式的起始位址，在進行插斷處理前，6502 處理器將先"獲取"這些位址。這些位置分別是：位址 FFFAH與 FFFBH（NMI），以及位址 FFFEH與FFFFH（IRQ）之記憶位置。這些位置由於實際都在含BASIC解釋程式（interpreter）與監督程式的唯讀記憶器內，因此，這四個位置的位址是固定的，您永無法加以改變。不過，這些固定位址僅指至其它可讀寫位置罷了，您實際上還是可由這些可讀寫位置上改變插斷服務副程式（interrupt service subroutine）的指示器。有關這些"向量"位置的用法，詳情請看"**Apple II 參考手册**"。

4－9 結　語

結束本章之前，最後我們還需說幾句話。本章，我們曾

介紹了一個對兩個位元組做邏輯 AND 或 OR 運算的簡單組合語言副程式，以及組合語言副程式叫用指令 USR 的用法。實際上，Apple 計算機之指令集本身即含有一旗號檢查命令：WAIT。這個指令可用以檢查每一個別旗號或一組旗號，同時，其亦可測知邏輯 1 與邏輯 0 旗號。但是，其用法亦有限制。倘若測不到適當的旗號內含，則程式將離開不了旗號檢查作業。此時，爲了重新獲得控制，您必須令計算機重置（ reset ）。同時，您亦無法在旗號（某一或某幾個）值爲 1 時，令控制跳至某一程式部份，而在旗號值爲 0 時令控制跳往另一個方向。若使用 WAIT，那您只有繼續 WAIT（等候）到條件滿足爲止。由於這樣非常沒彈性，因此，我們一直避免使用 WAIT 命令。

若您擴大視野，進一步學習組合語言程式設計，您會發現，叫用組合語言副程式的 USR 命令極有價值。不過，若您僅想叫用一個諸如邏輯 AND 運算的副程式，那您可使用 CALL　指令，並在這個指令之後附上副程式之十進起始位址。譬如，CALL　771　即可叫用前面的邏輯 AND 副程式。當然，在叫用之前，您仍然必須將面罩與資料位元組 POKE　入適當位置。

本章的目的乃在向您介紹一點 Apple 計算機的威力，以及其如何處理不同的工作。相信您有感覺，易途並不總是最有趣或最富教育性的。

5

Apple 之麵包板

人們一直認爲，不論在程式開發或硬體（界面）開發上，計算機應儘量易於使用。由於大多數計算機界面所需的信號在計算機系統上都有，因此，我們乃決定開發一些可用於許多不同計算機上的通用界面電路。這些電路非常簡單，極易製作，且適用於含 Apple 在內的許多計算機。我們同時亦設計了一個可包容界面所需之全部電路的印刷電路板。這個界面電路的照片即如圖 5 － 1 所示。界面麵包板與各計算機間以 40 號的平面電纜連接。雖然這個界面電路可事先裝好在麵包板上，然後再用來做實驗，但我們認爲，這樣只有增加問題罷了！

5 － 1　基本麵包板

基本麵包板上含有許多能讓您輕易製作與測試界面設計的電路。這些電路包括電源、邏輯探測器、設備與記憶器解碼器、巴士緩衝器、以及控制電路。

電　源

麵包板的電源部份可有兩種不同的製作方式　若能供應1安培的電流，則您可外加一正五伏特的電源，否則，你就必須使用一外接變壓器，供應交流 12.6 伏特的電壓至麵包板上的電源電路。不論用那一種方法，麵包板上的電源都是

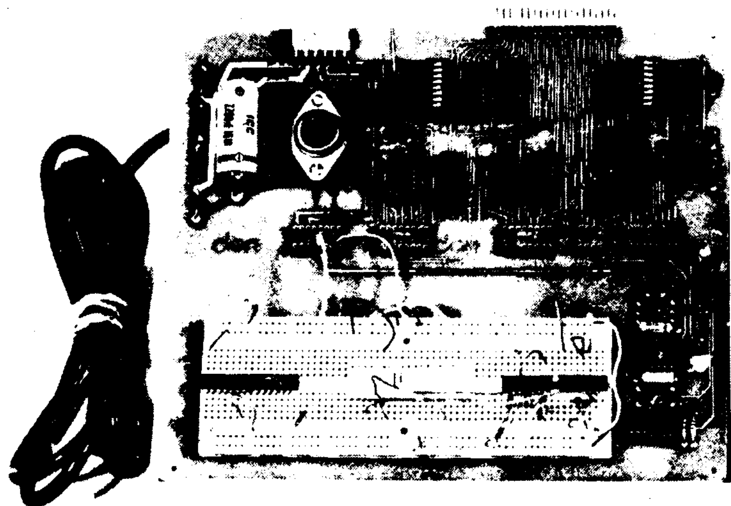


圖 5・1　Apple麵包板系統

與計算機的電源分開的。此乃由於若干計算機系統無法同時供給足夠的電力給自己，以及給您所欲測試的外加界面電路，因此，界面電路必須另外使用一個電源。不過，在使用外加電源時，您隨時都要記得兩個電源之間必須有良好、低阻

抗的共同接地。圖5－2所示即為電源電路的線路圖。

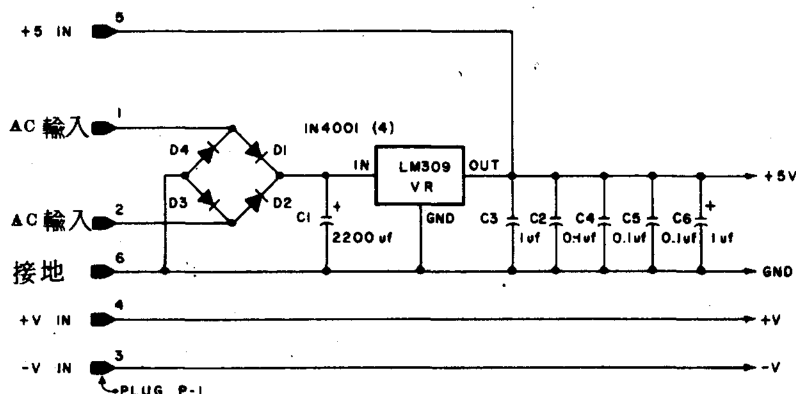


圖5－2　麵包板電源電路的線路圖

採用基板上的電源時，12.6伏特的交流變壓器必須接在1號插孔（P1）的第1與第2接脚上；而整流子D1～D4、濾波電容C1、以及電壓調整鈕VR則都裝在裡面。+5伏特的調節器上最好能加一個小散熱片。麵包板這樣使用時，+5伏特在P1的第5接脚上，而接地在第6支接脚上。必要時，這些接點可接外部設備。

若另外採用一個+5伏特的電源，則電源零件D1～D4，C1與CR等就不需要了，而且應除去或拆掉。這時，+5伏特與接地同樣分別在P1的第5與第6支接脚上。

由於經常還會用到其它電源，如±12或±15伏特，因此，我們特別設置了P1接點，以便能連接其它的外部電源。正電壓（+V）與負電壓（－V）分別接在P1的第4與

第 3 接脚。

所有的電壓均可由 IC－16 位置的插孔上獲得，這些現成的接點如表 5 － 1 所示。

表 5 － 1　IC－16電源插座上所有的電源接點

接脚*	可得電壓
7,10	+5
5,12	GND
3,14	+V（外部）
1,16	−V（外部）

* 所有其它接脚都不接

印刷電路板上之 IC 的電源均取自＋ 5 伏特電源。 IC－16 插座上的接點使您能輕易地獲得實驗所需的電源。

邏輯探測器

圖 5 － 3 所示的邏輯探測器電路在求知各種輸出的邏輯準位，以及在測知輸出端之脈衝活動上十分有用。麵包板上之邏輯探測器部份含有一個準位測知器以及一個脈衝測知電路。準位測知器以一 LM－319（ IC－15 ）比較器測知邏輯 1 與邏輯 0 準位，而脈衝測知電路則以一 SN74LS123（ IC－14 ）IC 測知與“展開”脈衝。我們同時亦分別以一綠色發光二極體（ LED ）作邏輯 0 指示器（ D－7 ），一紅色 LED 作邏輯 1 指示器（ D－6 ），以及一黃色 LED 作脈衝指示器（ D－5 ）。探測器的輸入在 IC－19 IC 插座之

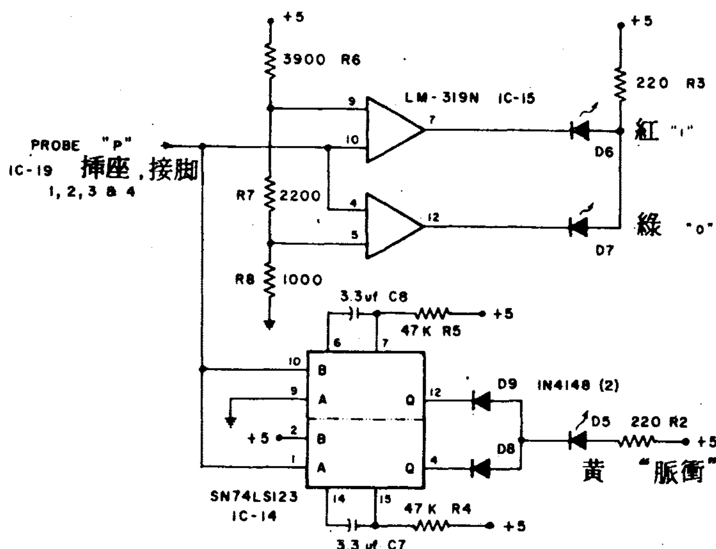


圖 5－3　邏輯探測器電路圖

第 1 ～ 4 接脚上。這些輸入上都標有“ P ”，其全是並聯的，使用任一者都可以，但切勿將邏輯探測器同時接至兩個信號上。邏輯探測器相當於兩個低功率的蕭克力（ L S ）輸入負載。

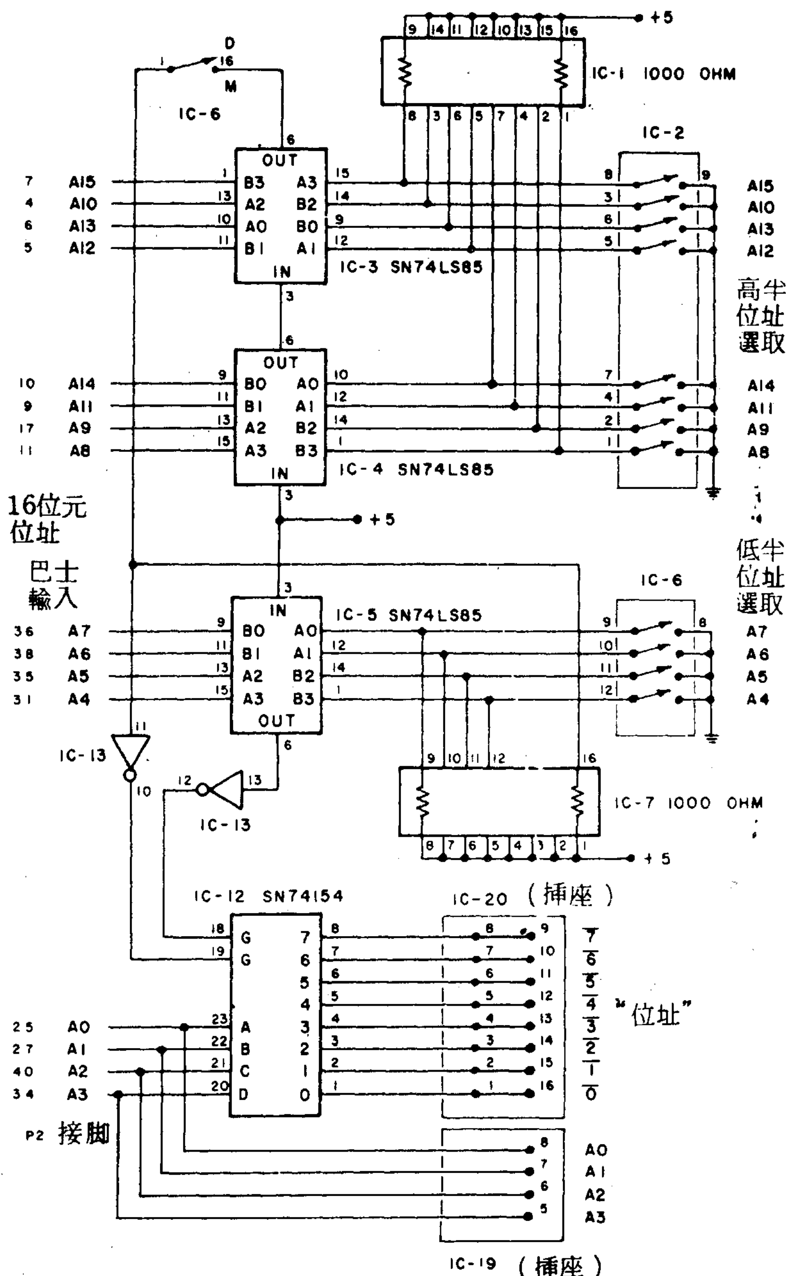
若您自己擁有邏輯探測器，則這一部份的電路就可以不要了。願意的話，您亦可不必裝置這一部份電路。不過，能測知脈衝，並且測知脈衝的狀態還是很有用的。經驗已經證明，這種邏輯探測器在麵包板界面電路的檢修上非常有用。

記憶器與設備解碼器

麵包板上之電路有一個主要部份是專作輸入/輸出位址解碼用的。這個電路正如圖 5 － 4 所示。依正在使用中的不同計算機型態而定，這個解碼器可動作於設備型態，亦可動作於記憶器型態。設備選取僅解碼下半部的位址位元（ A7 ～ A0 ），而記憶器選取則解碼全部的位址位元（ A15 ～ A0 ）。由於採用 6502 微處理器構成，因此，Apple 計算機以記憶器定址選取輸入/輸出設備。以 6800 微處理器所構成的計算機同樣亦採用記憶器定址選取輸入/輸出設備。不過，以 8080 、 8085 、或 Z80 微處理器所構成的計算機，則兩種型態都可使用。仔細看看圖 5 － 4 之電路圖，您即可知道，位址解碼混合使用了數位比較器與解碼器。

設備定址以一 SN74LS85 四位比較器（ IC－5 ）將預先設定好的位址位元與下半位址巴士線 A7 ～ A4 上所呈現的位址位元相比。 IC－6 處的開關就是用以設定這些欲與位址巴士比較的邏輯準位。 IC－6 處的元件是一組雙列並排的開關，因此，在做開關設定時必須特別小心。這些開關位置在" LO "之開關處都清楚標上" 7 "、" 6 "、" 5 "、與" 4 "的記號。在安裝這些開關時，特別記住，關的位置在右邊（邏輯 1 位置）。當開關置於邏輯 1 的位置時， IC－7 處的提升（ pull-up ）電阻對 SN74LS85 提供邏輯 1 輸入。

當位址巴士位元 A7 ～ A4 與預設位元吻合時，



+5
D
M
IC-6
IC-1 1000 OHM
IC-2
OUT
IN
A15
A10
A13
A12
IC-3 SN74LS85
高半位址選取
A14
A11
A9
A8
IC-4 SN74LS85
16位元位址巴士輸入
低半位址選取
IC-5 SN74LS85
A7
A6
A5
A4
IC-13
IC-7 1000 OHM
IC-12 SN74154
IC-20 (插座)
"位址"
A0
A1
A2
A3
P2 接腳
IC-19 (插座)

圖5－4　位址解碼電路圖

SN74154 解碼器（IC－12）即致能。雖然SN74154 能將A3～A0等位址位元解碼成16個獨特的位址輸出，但這個電路僅使用了前面八個——這樣即已足足有餘麵包板及界面測試用了。

所以，若A7～A4位址位元的開關設定爲1011，則解碼器所解碼的位址即爲10110000_2至10110111_2，寫成十進制則爲176至183。在設備定址時，IC－6的最下面一個開關必須置於“打開”或“D”的位置。這樣才可使解碼器動作於正確的型態。

解碼過的位址輸出呈現於IC－20插座上。這些輸出分別標有“0”、“1”、……等等一直至“7”。這整個部份稱爲“ADDRESS”（位址）部份。注意，每一位址數上都加有一短槓，以代表每一獨特輸出的狀態均爲一邏輯0脈衝。0至7之位址記號正好代表著一種循序的位址，以幫助您得知那一支接脚連接在設備位址輸出上。但讀者應記得，絕大多數情況下，這些數目和經解碼過的實際位址間都無關連。因此，就前面我們所舉，解碼176至183之位址而言，標爲“0”的輸出即正好代表位址176之解碼輸出，標爲“1”的輸出即代表位址177的解碼輸出，如此類推。表5－2所示即爲IC－20之位址插座上的解碼輸出詳情。

記憶位址在界面麵包板上亦極易解碼。我們可以另兩個比較器晶片：IC－3與IC－4，將A15～A8之位址位元與一事先設定之高（HI）位址相比。這些高位址位元以IC－2上稱爲HI之雙列並排開關元件中的八個開關設定。在使用記憶器選定時，您必須特別小心，千萬不能用到已經分

表 5－2　IC－20　位址插座上之位址解碼器接點

接脚 (IC-20)	記　號	SN74154 輸出接脚
1,16	0	1
2,15	1	2
3,14	2	3
4,13	3	4
5,12	4	5
6,11	5	6
7,10	6	7
8,9	7	8

配給 Apple 內部記憶器（ROM或R/WM）的位址。您還必須記得，在使用 PEEK 與 POKE 指令時，16 位元的位址必須轉換成十進數。

爲了使電路動作於記憶器選定型態，IC－6 最下面一個開關必須置於“閉”或“M”的位置。此舉使得 SN74154 解碼器唯有在一種情況下才動作，那就是高半開關所設定之位元值與 A15 ～ A8 之位址位元吻合，而且低半開關所設定之位元與 A7 ～ A4 等位址位元亦吻合時。如此，記憶器選定所能存取的位址即爲××××××××　××××0000 至××××××××　××××0111 之間，其中，×代表 0 或 1 均可。這些經解碼過的位址以邏輯 0 脈衝的型式呈現在“ADDRESS”插座（IC－20）。記得，每一被選定的 16 個位址中，僅有最前面 8 個位址可用。因此，若高半位址預設值爲 10000001，且低半位址（A7 ～ A4）預設值爲 1110，則 33248 ～ 33256 幾個位址將分別在“ADDRESS”插座的第 1 至 8 接脚上產生一邏輯 0 脈衝。記住，SN74154 解碼器解碼了全部 16 個位址，但我們僅用了其“下半部”8

個。

A3～A0 等位址線（未加緩衝）的接點分別在 IC－19 插座之第 8 至 5 接脚上。這些信號還是可用在一些實驗中，不過，使用時必須特別小心，因為，這些信號未經緩衝，而直接與 Apple 計算機連接。

麵包板的位址解碼器節省您不少時間與精力，因為，有了這個，當您希望製作輸入／輸出口或嘗試一些簡單的界面電路時，您就不必再製作設備位址解碼電路了。

巴士緩衝器

正如圖 5－5 所示，巴士緩衝使用了兩個 8216 非反相

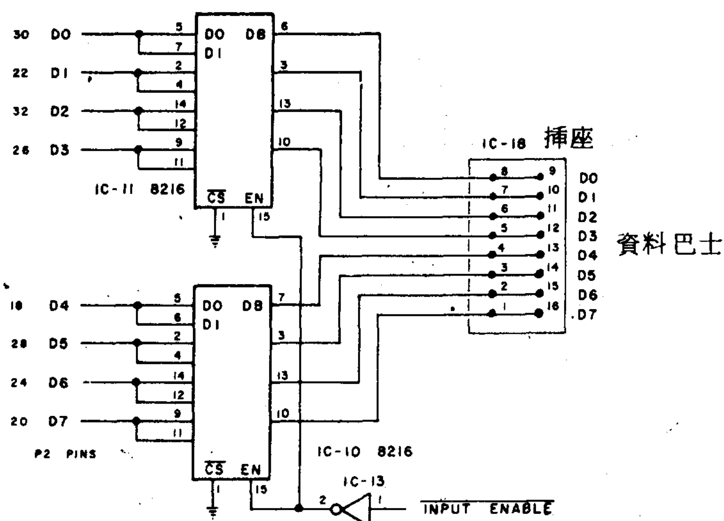


圖 5－5 巴士緩衝電路圖

巴士緩衝器晶片IC−10與IC−11。這個意思是說，巴士的總輸出量（fan-out）達30（亦即，其可推動30個標準的7400-型輸入），而且其與Apple的資料巴士隔離。資料巴士的八個位元接點均在IC−18插座上。表5−3之訊息說明了這個資料巴士的接脚情形。

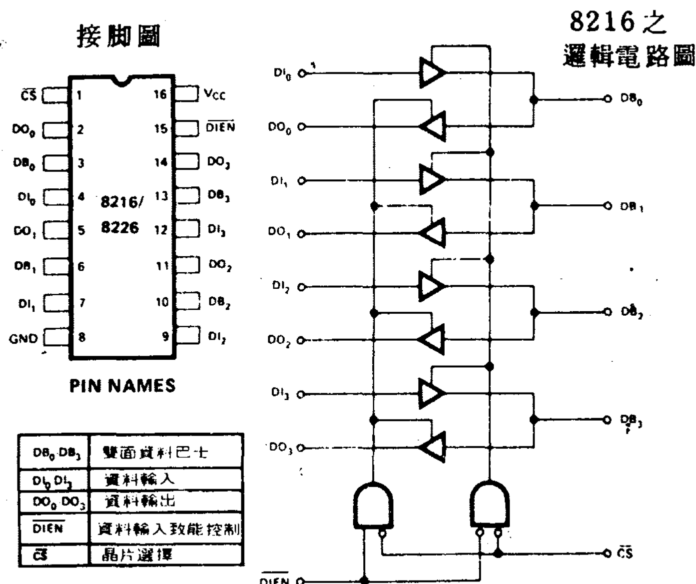
表5−3　IC−18插座上之資料巴士接點

接脚(IC-18)	資料巴士信號
1,16	D7
2,15	D6
3,14	D5
4,13	D4
5,12	D3
6,11	D2
7,10	D1
8,9	D0

巴士緩衝器永遠致能，而且正常的作業型態爲將資料由Apple傳遞至麵包板。這意思說，只要將邏輯探測器或其它合適的監聽器接至巴士緩衝器晶片的輸出D7～D0，不必再使用其它信號，您即可監視到巴士的“活動”。而輸出口的製作就是藉適當的控制信號（在下一節介紹）控制一個八位元的鎖住器。這八個鎖住器輸入即接至IC−18插座的D7～D0。

不過，輸入口必須製作成能以反方向推動巴士緩衝器，以便能將資料“驅入”Apple。實際上，如圖5−6之8216緩衝器所示的，每一巴士線都有兩個巴士緩衝器。在圖中，DIEN輸入決定那一組緩衝器應致能，以使資料流入

或流出 Apple 。所有的輸入作業都必須使適當的一組緩衝器動作，以使 Apple 能正確地收到資料。因此，輸入作業均有特殊的控制電路來做這個。



DB_0-DB_3	雙面資料巴士
DI_0-DI_3	資料輸入
DO_0-DO_3	資料輸出
$\overline{DIEN}$	資料輸入致能控制
$\overline{CS}$	晶片選擇

圖 5－6　8216巴士緩衝器晶片之接脚圖

控制電路

麵包板上的控制電路相當簡單，主要包括一些通用的緩衝器，作計算機所輸出之控制信號的緩衝。這些控制信號總共有六個： $\overline{IN}$ 、$\overline{RD}$ 、$\overline{OUT}$ 、$\overline{WR}$ 、$\overline{RESET}$ 、與 $\overline{INTAK}$ 。就 Apple 界面而言，您只需用到 $\overline{WR}$ 、$\overline{RD}$ 、與 $\overline{RESET}$

三個信號就可以了。其它的信號只有在麵包板與其它計算機合用時才用得上。圖 5 － 7 所示即爲這個控制電路。

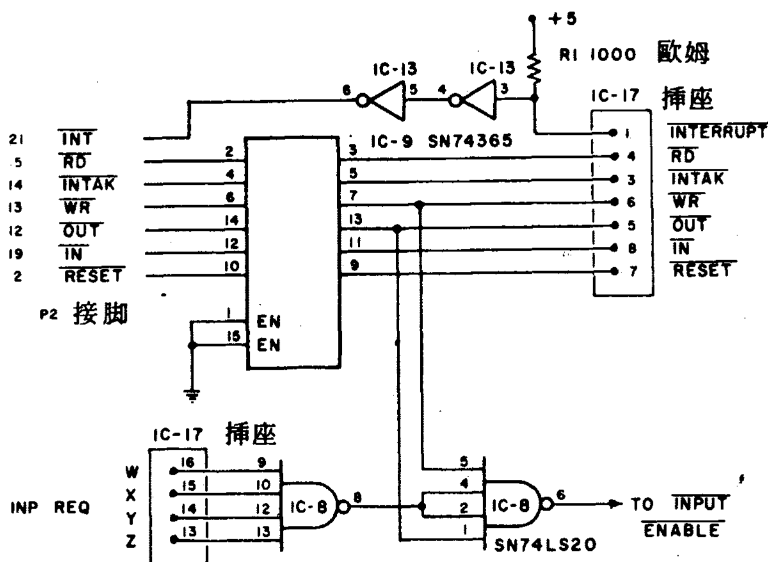


圖 5 － 7　控制電路圖

由圖中您可看出，通用的插斷信號亦經過緩衝，不過，其乃一計算機之輸入。如表 5 － 4 所示，控制信號接點均做在

表 5 － 4　IC －17處之控制信號接點

接脚(IC-17)	控制信號	方向
1	$\overline{\mathrm{INT}}$	輸入
2	不　用	—
3	$\overline{\mathrm{INTAK}}$	輸出
4	$\overline{\mathrm{RD}}$	輸出
5	$\overline{\mathrm{OUT}}$	輸出
6	$\overline{\mathrm{WR}}$	輸出
7	$\overline{\mathrm{RESET}}$	輸出
8	$\overline{\mathrm{IN}}$	輸出

IC－17 插座內。

這個控制電路亦產生一個將 8216 巴士緩衝器轉換成輸入型態的信號，以使資料能送入Apple。當發生記憶器讀取作業時，表面上看來似乎只需將巴士回轉一下就夠了。然而，假設眞的這樣做，則甚至當Apple內之記憶晶片在動作時，麵包板上的巴士緩衝器都會變成輸入型態。這就造成了巴士“衝突”。因此，麵包板上的巴士僅能在麵包板上的輸入設備本身已被選取時，才能變成輸入型態。

爲了正確地處理輸入口，我們以輸入口之設備選取信號將資料閘控至資料巴士上，並且控制了 8216 巴士緩衝器的作業型態。實效上，最多總共有四個輸入口的設備選取脈衝可彼此“ OR ”在一起，以將麵包板的巴士緩衝器置於輸入型態。由於在麵包板上您不可能用到四個以上的輸入口，因此，這些信號唯有在當麵包板上產生一輸入口設備選取信號，而且用者將這個信號接至四個巴士緩衝器的致能輸入其中之一時，才令巴士倒轉，輸入資料。

“ INP REQ”（輸入請求）控制脈衝一定要爲邏輯 0 脈衝。這些脈衝加至 IC－17 插座之 16 至 13 接脚上，這幾支接脚分別稱爲W、X、Y、與Z。

這些控制信號的實際OR運算則由 IC－8 、SN74LS20 閘完成。這個四輸入之 NAND 閘所輸出的輸入請求信號，進一步又與 $\overline{\text{OUT}}$ 及 $\overline{\text{WR}}$ 閘控在一起。這個閘控主要在作安全鎖住，以使在麵包板電路萬一有接錯的情況下，發生輸出作業時，巴士推動器不致於被設定成輸入型態。這個閘控所產生的信號控制了 8216 巴士緩衝器的輸入／輸出型態。

由於Apple僅產生記憶器寫入信號$\overline{WR}$，因此，當計算機正在執行寫入動作時，您的界面根本無法令巴士倒轉做輸入動作。$\overline{OUT}$信號用於與8080、8085、或Z80計算機之界面。

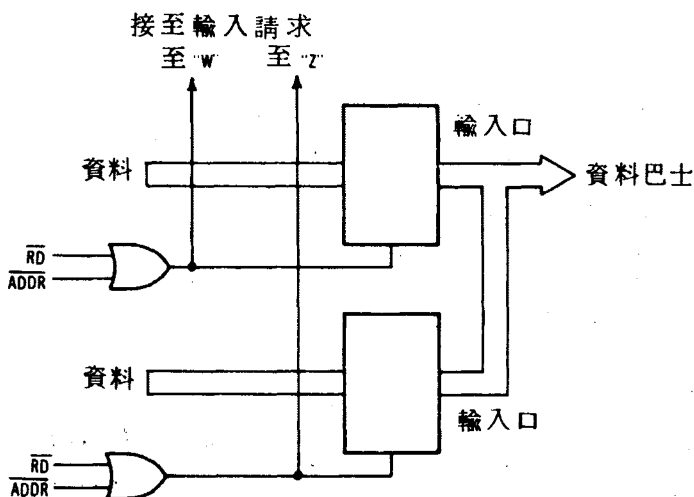


圖5－8　顯示輸入請求信號之用途的典型輸入口

圖5－8顯示了兩個輸入口。其中每一個輸入口均受一令三態緩衝器致能之設備選取脈衝的控制。這個信號同時亦被用作輸入請求信號 INP REQ ，而且每一輸入口必須產生其各自之輸入請求信號。就這個例子而言，兩個輸入請求信號已分別連至 IC－17 插座之 INP REQ 部份的W與Z接脚。這些線連至X與Y接脚亦同等容易。

鎖住之輸入請求信號與相關電路之使用，唯有在測試麵包板上的界面電路時才能使用。倘若您所欲製作的是一個欲

直接插入Apple，而且不使用雙向巴士緩衝的界面，那您就不需使用這樣一個鎖控。這個電路的主要目的乃在保護您的Apple計算機，防止其受到因不小心或不正確接錯測試電路所造成的傷害。一旦電路都完全測試過且除過錯，則您即可放心地直接將之接至Apple之資料巴士上。

麵包板接線

麵包板電路可以如圖5－9所示的包線技巧製作而成。在這種情況下，只要將接線做簡單的改變，電路即可擴充或修改。不過，麵包板本身可能有一點難用。

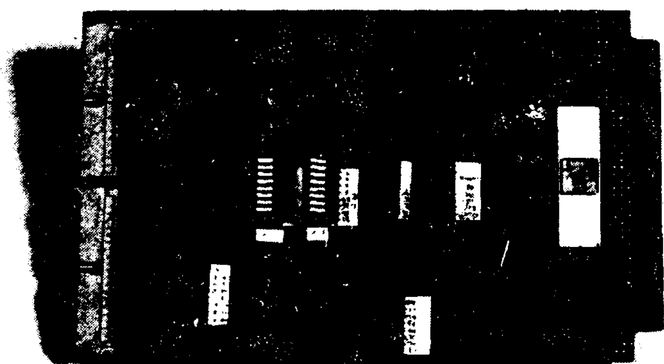


圖5－9　界面電路之包線型式

為了幫助您製作與測試界面，我們特別設計了一塊能包含所有必要電路的印刷電路板。為了使麵包板易於使用，電

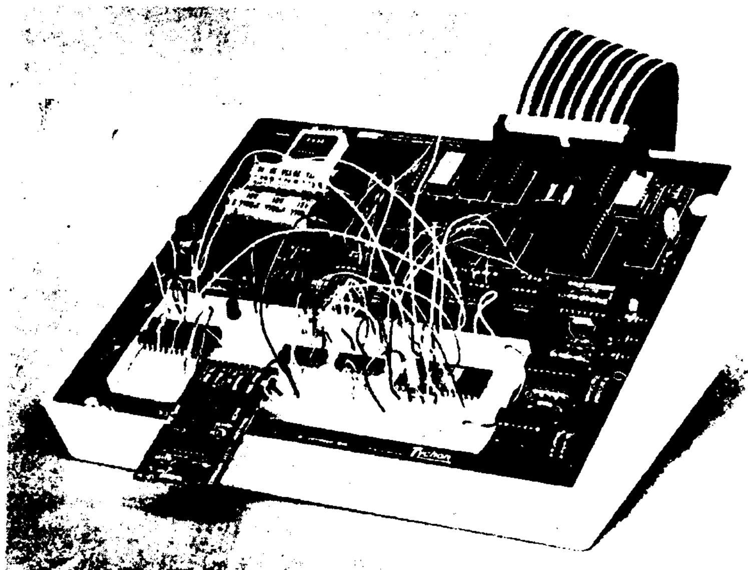


圖5－10　界面之封裝型式

源與邏輯探測器電路亦包括了在內。這個麵包板即如圖5－10所示。若有需要，可直接寫信到 Group Technology, P.O. Box 817B, Check, VA 24072 訂購，有套件型式，亦有組合好的型式。為了便於做實驗，麵包板上特留了很大的空間未用，以讓您能在印刷電路板上直接裝上一免焊式的麵包板插座。典型的麵包板插座有E & L Instruments公司（地址：Derby CT 06418）的"SK－10"，以及AP Products公司（地址：Mentor, OH 44060）的"Super Strip"。製作麵包板電路所需的全部零件，以及印刷底板的線路設計均附於附錄中。

5－2　接到Apple上

由於界面麵包板以 40 號之電纜接至各種計算機，因此，您需有一種方法將電纜連接至Apple內之其中之一週邊界面槽。作者在此建議您使用如圖 5－11 所示的平面電纜組合線。電纜線之一端有一印刷電路板邊緣母接頭，且另一端有一0.1 英吋乘 0.1 英吋之母柵接頭。兩個接頭的開口必須面對同一方向。

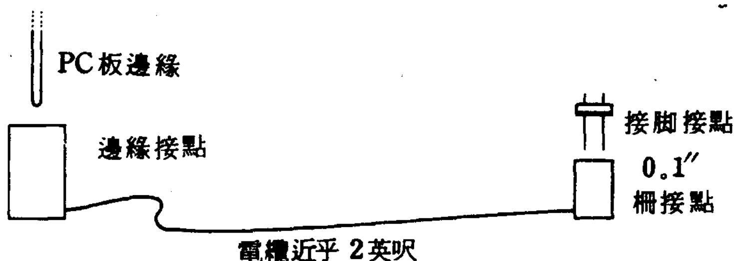


圖 5－11　界面電纜

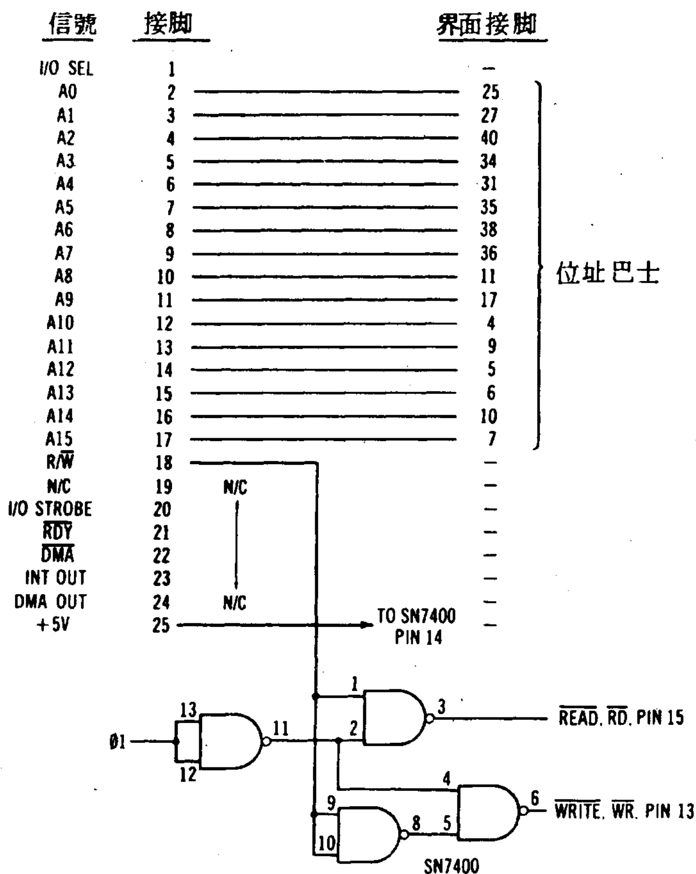
實際與Apple巴士信號之連接則以一小調適板達成。這調適板“扭曲”與“轉動”各種信號，使其沿著由邊緣接點至Apple內之週邊接點的路線傳遞。調適器可以 Vector 4609 模型板輕易地湊在一起。調適板插入Apple內之其中一週邊接點，而且其具有一將直接連接至界面電纜的 40 號導線邊緣接點。當然，若您願意，您可直接在電纜上焊接，

但作者建議您最好不要這樣做。您可利用短小的掛鉤線，在每一邊緣接點的對應信號導體上直接焊接。倘若您不願意用焊接，則您可將包線接脚焊在每一邊緣接點的孔內，以環包線作連接。

實際的接線情形即如圖 5－12 所示。若您選擇使用 Vector 模型板，則不論您採用那一種接線技巧，在開始將兩邊緣接點連接前，有許多重要的事您必須先做好。印刷電路可能有一、兩處脫落，或者，40 號接點以及＋5 伏特與地之間的導電路徑，接觸到 50 號的 Apple 接點。像這些接觸都必須弄斷，以使 40 號的接點能“獨立”，且不託付於任何信號。這些不正當的連接您可以剃刀將之刮去。我們建議您每一接線刮兩道，各離約 2 至 3 毫米左右。然後，再以烙鐵燒燙刮斷的部份，讓導體脫落。這只需針對連接兩接點間的電源接線即可，其它的所有接脚都是“獨立的”。

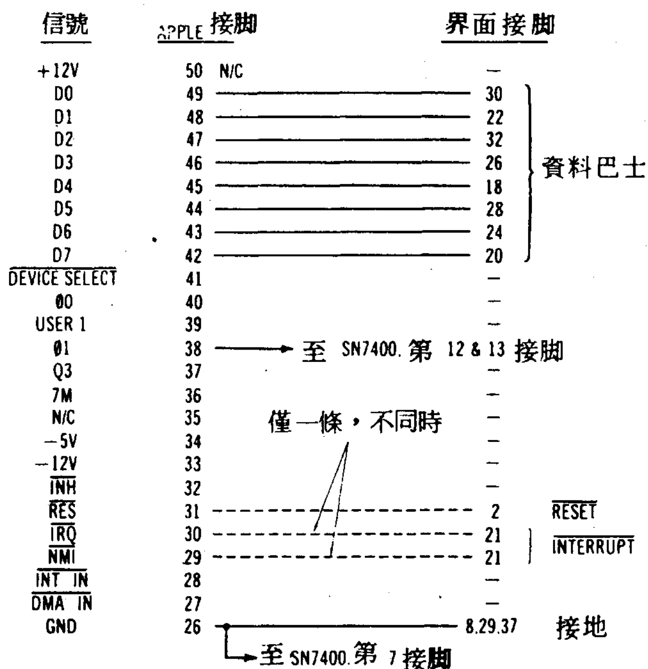
由於 Vector 模型板不用平穿洞，因此，記得＋ 5 伏特與接地一定要接至各自的電源巴士上，而且，SN7400 晶片的連接一定要正確。

SN7400 晶片用以閘控讀寫（ R/$\overline{W}$ ）信號與 6502 微處理器的主時序信號 $\phi 1$ 。這個閘控產生記憶器讀取信號 $\overline{RD}$ 以及記憶器寫入信號 $\overline{WR}$ 。倘若不做這個閘控，則界面麵包板上的計算機週邊將無法正常動作。在有些計算機內，讀取與寫入信號是分開的。在 Apple 以及在其它以 6502 處理器所組成的計算機系統內，若您希望以分開的讀取與寫入信號作記憶器控制，那您就必須以適當的閘控自己產生這些信號。



信號
接脚
界面接脚
I/O SEL 1 —
A0 2 25
A1 3 27
A2 4 40
A3 5 34
A4 6 31
A5 7 35
A6 8 38
A7 9 36
A8 10 11
A9 11 17
A10 12 4
A11 13 9
A12 14 5
A13 15 6
A14 16 10
A15 17 7
位址巴士
R/W 18 —
N/C 19 N/C —
I/O STROBE 20 —
RDY 21 —
DMA 22 —
INT OUT 23 —
DMA OUT 24 N/C —
+5V 25 TO SN7400 PIN 14 —
Ø1
13
12
11
1
2
3
READ, RD, PIN 15
4
5
6
WRITE, WR, PIN 13
9
10
8
SN7400

圖 5 —12 Apple



信號
APPLE 接脚
界面接脚
+12V 50 N/C —
D0 49 30
D1 48 22
D2 47 32
D3 46 26
D4 45 18
D5 44 28
D6 43 24
D7 42 20
資料巴士
DEVICE SELECT 41 —
Ø0 40 —
USER 1 39 —
Ø1 38 至 SN7400. 第 12 & 13 接脚
Q3 37 —
7M 36 —
N/C 35 —
−5V 34 —
−12V 33 —
INH 32 —
RES 31 2 RESET
IRQ 30 21
NMI 29 21
INTERRUPT
僅一條，不同時
INT IN 28 —
DMA IN 27 —
GND 26 8.29.37 接地
至 SN7400. 第 7 接脚
---- = 可有可無之接線

至界面接點的連接

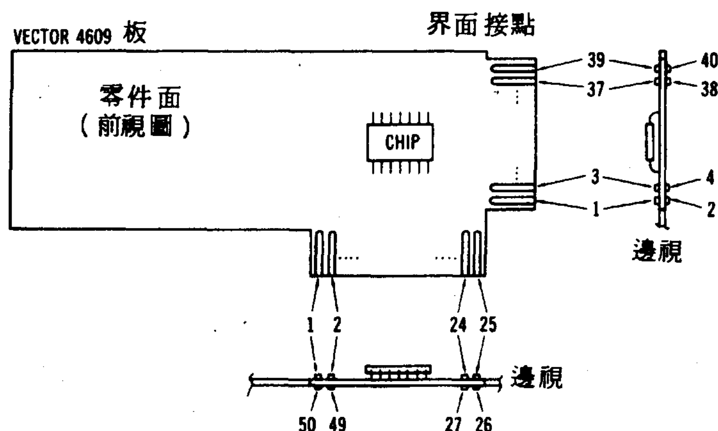


圖 5－13 Vector 4609 板接點與界面接點之安排

Vector 4609 板邊緣接點的接脚位置示於圖 5－13 中。請留意，這個圖畫的是 Vector 板的零件面。在做好兩邊緣接點以及接點與 SN7400 間的連接後，我們建議您以一歐姆表或其它連續檢查的儀器，確實檢查一下，看鄰近與對面接脚之間是否有短路情形發生，以及接線是否都正確無誤。做這些測試時，SN7400 晶片必須拔離插座。測試完後記得再將之插回去。

5－3 其它考慮事項

若您願意分別嚐試一下一些 6502 族的界面晶片以及一些非族系內的晶片，您會發覺，比起諸如 SN74365 與

SN74LS244 等等之標準三態輸入晶片，這些晶片的存取速度實在相當慢。實際上，這些大型可程式化晶片的存取時間可能長達 200ns 。而由於 6502 晶片的讀／寫時序要求十分嚴格，因此，若將 8216 巴士緩衝器晶片以及鎖住電路的額外延遲時間計入，這些晶片可能沒有足夠的時間存取資料並將資料置於巴士上。因此，倘若您想以麵包板測試使用複雜可程式化界面晶片之界面電路，您則必須“克服”鎖住。這只需拿掉兩個 8216 巴士緩衝器晶片，並且在每一插座以短短的跳線將Apple的資料巴士信號與界面資料巴士線連起即可。舉個例子而言，每一插座之第 5 與第 6 ，第 2 與第 3 ，第 13 與第 14 ，以及第 9 與第 10 接脚之間，都必須有一跳線連接。希望您參考圖 5 － 5 使用 8216 巴士緩衝器晶片的電路。

不過，有句話要提醒的。在除去巴士緩衝器晶片後，界面電路就直接與 Apple 計算機的資料巴士接在一起了。因此，在做這個時，希望您極度地小心，切勿造成短路或 Apple 內之巴士衝突。第 7 章裡我們將舉一個使用直接巴士界面的簡單例子作參考。

6

Apple 界面實驗

這節實驗的目的是在讓您親自動手使用前面幾章所討論過的鎖住輸出口以及三態輸入口電路。您會發現，這些實驗以簡單的SN7400系列元件，將資料傳入與傳出Apple。

6－1　實驗簡介

這一章必須將電路裝在麵包板上，完整的零件單摘列於附錄B中。本書假設您已有將簡單邏輯電路裝在麵包板上的經驗，並假設您熟悉基本的插麵包板技巧。實驗中將用到一些監聽邏輯狀態與產生邏輯狀態的輔助功能。通常，我們都可以燈泡或 LED 顯示邏輯1（亮）與邏輯0（暗），以開關產生邏輯準位，並以除跳訊脈衝器（簡稱脈衝器）產生邏輯準位轉態時無雜訊的邏輯準位。附錄中附有一點這些電路的簡單電路圖。若您不願意製作這些電路，則您可個別將之裝在麵包板上。一般而言，這本書的絕大部份實驗都可以簡單的電路完成。

我們舉了一個以解碼器電路作設備選取的實驗例子。雖然解碼器方法有許多，但我們認爲就一個實驗即可闡明基本的原理。若您對其它的電路還有興趣，則請您參考諸如"8085A 手册"以及"6502 程式設計與界面實驗"等有關書籍。事實上，各種計算機間之記憶器與輸入／輸出設備選取則是大同小異的。就絕大多數界面電路而言，界面麵包板上所用的解碼器電路就已經相當夠用了。

雖然這本書以一相當低的層次處理Apple 界面，但您可能還希望了解一些其它重要的界面主題。這些主題在" **TRS-80 界面實驗，第二册**"幾乎都有談到。有興趣的讀者希望您參考這本書。這本書上所討論的東西都很一般化，因此，其極其容易應用至Apple 計算機系統上。這些主題包括：高電流、高電壓負載推動、數位至類比以及類比至數位轉換器，實用資料處理（平滑、濾波、平均、等等），串聯通信，與遙端控制。

圖 6-1 所示即爲本章之實驗所使用之Apple 計算機與麵包板的照片。麵包板與Apple 計算機之間以如圖 6-2 所示之 40 號電纜連在一起。這個電纜在第 5 章時已介紹過。在將界面麵包板接至Apple時，特別記住，電纜的取向一定要對。**電纜恒背對將界面連接至 Apple之調適板的零件面**。在電纜之界面一麵包板端，電纜必須推入 40 支接脚，以使電纜面向下或背向印刷電路板。假若電纜沒接好，則 Apple將在螢幕上顯示出一堆雜亂文字，而不是像電源一打開時所顯示的Apple Ⅱ 標幟。這種錯接只要不要維持太久，Apple或界界面似乎不致造成任何永遠的傷害。

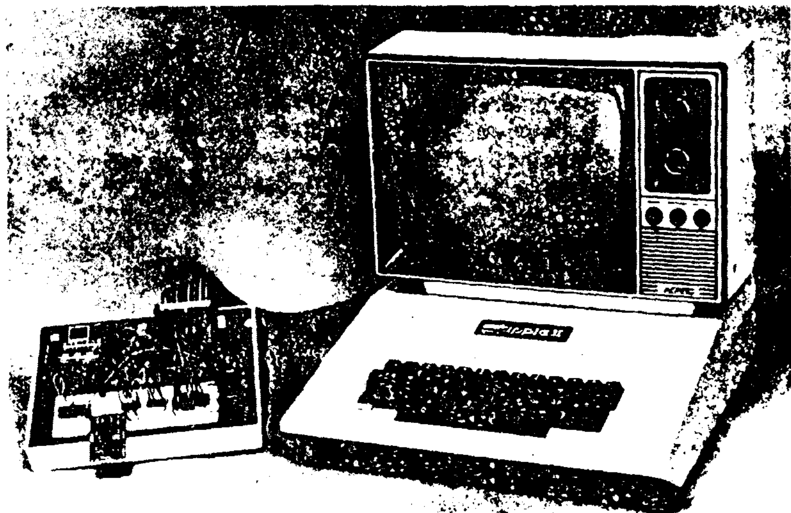


圖 6－1　實驗中所用的Apple計算機與麵包板。



圖 6－2　界面電纜(注意接點方向在平軸電纜同一邊)。

有部份實驗會植基於或使用前面實驗所發展出的電路或程式。除非有教您這樣做，否則，請勿隨意將計算機的電源關掉。不然的話，您就得花費許多時間重新取入程式或重新裝配界面電路。有些實驗後面我們會加上諸如此類的提示語。

相信絕大多數讀者都會循序地做這些實驗，在這種情況下，實驗參考到前面的地方就沒什麼問題。倘若您是跳著做，有時您就會覺得有些東西不甚了解。爲了幫助您了解這些界面電路，在本章末了，我們特別又將最重要的輸入口、輸出口、與控制電路畫出於圖 6－27 中。爲了能隨手參閱，您可將這張圖複印一份，或直接將這幾頁拆下，放在手邊。除非特別聲明，否則，本章之絕大多數實驗所用的都是這張圖中所示的基本電路。必要時，您亦可以這些電路做成通用的輸入與輸出口。

假若您是老師，而且希望以這本書作 Apple 計算機的實驗教本，則您會發現，書中的程式很容易即可存在卡帶上。將程式存在卡帶後，學生即可隨手獲得這些程式，而不必浪費時間爲程式除錯。若您選擇使用卡帶，一定要使用高品質的卡帶，而且程式一旦錄在磁帶上後，卡帶背緣之"寫入保護"鈕就應拿掉。因爲，這樣可防止學生因意外寫入磁帶而破壞原有的程式。

學生或許發現自己亦應擁有一份卡帶程式，以便能隨時獲得實驗解與其它程式，不管是和其它人作交換或留作爾後參考。

本章的實驗已分成兩組，雖然沒有明顯的章節區分，但由許多小標題以及其它之註解，您將可發現這種分野。其中

，前 11 個實驗所做的都是基本界面與程式設計技術的探討。這幾個實驗可爲初步的計算機界面或計算機電子學課程的實驗部份，提供一個基礎。

最後幾個實驗則對幾個更高深的課題作實驗的探討，其同時亦提供了可用以更充實前面之基本實驗的研究計劃。當然，所有這些實驗還是都可以做。

6－2　實驗

實驗 1

邏輯探測器之應用

目的

這個實驗的目的是要告訴您如何以裝在麵包板上的邏輯探測器電路，測知邏輯準位與脈衝。

討論

假設您所用的都是裝在麵包板上的邏輯探測器，當然，其它的邏輯探測器亦同。本實驗的步驟可幫助您熟悉麵包板

與現有可用的信號。

第 1 步

將Apple計算機接至視頻顯視器，且經 40 條導線之電纜接至界面麵包板。這個接線在實驗簡介時已說明過。

打開Apple計算機與麵包板的電源。計算機螢幕上應出現" APPLE Ⅱ"的字樣以及一閃爍的方形光點(square cursor)。萬一沒有，則先關掉電源，仔細檢查一下接線，然後再重新開機。記得， 40 號電纜一定要安穩地插入界面麵包板之接脚內以及在接至Apple之板緣上。電纜的面向也應檢查一下看是否正確。若您自己一個人無法找出問題所在，找個人來幫忙。

第 2 步

電源加至麵包板後，以一條跳線將PROBE (探測器) 插座之其中之一邏輯探測器輸入接脚P，與電源插座處之其中之一 + 5 伏特電源接脚接起。看看邏輯探測器指示器會有什麼反應？

紅色 LED 亮，代表出現邏輯 1 狀態。

接著將跳線由+ 5 伏特之電源接脚移至同一插座之其中之一接地接脚。看看結果又如何？

綠色 LED 發亮，代表探測器電路之輸入出現邏輯 0 狀態。您可能已發現，在您將跳線接至＋ 5 伏特或接地剎那，脈衝測知（黃色）的 LED 會閃一下。這個閃亮即表示探測器已偵測到邏輯準位有所改變。不論遇到邏輯 1 變邏輯 0 或邏輯 0 變邏輯 1 ，黃色的LED都會閃亮一下。這個LED在測知脈衝與邏輯轉態上特別好用。

將探測器輸入接至 IC－19 的位址線 A0 。接好之後，您看到什麼呢？所有 LED 都發亮，但亮度或許有所不同。這主要是因為6502 微處理器晶片現在正在執行許多許多的組合語言指令並監聽著 ROM ，因而，正以位址巴士選取許多不同記憶位置。將邏輯探測器之測試線再移至另外其它位址巴士線A1，A2，A3……等等看看，您會發覺，在這些接脚上所測得的結果亦類似。事實上，不論那一條位址線，由於6502 微處理器均不斷有新的位址資訊送出來，因此，LED應迅速地不停閃亮著。

第3步

您或許希望以邏輯探測器測試麵包板上的一些其它點。譬如，資料巴士線與控制信號都可輕易地測試。不過，你要謹記，邏輯探測器僅接受麵包板及實驗中所使用之標準TTL所輸出的邏輯準位。因此，切勿以之測量這些準位以外的任

何東西。譬如，假若您將探測器接至一超出 0 至 5 伏特範圍的電壓，則探測器將會燒毀。

第 4 步

在使用探測器的過程中，您會發現， LED 發亮的情形有許多種組合情況。譬如，您可能看到紅色與黃色 LED 發亮，而綠色 LED 未亮。您知道這是什麼意思嗎？

這個意思是，探測器測到了一個脈衝，而且正被測試之電路的正常邏輯準位是邏輯 1 。綠色 LED 很快地亮一下（您看不到），顯示邏輯 0 脈衝疾逝的出現。脈衝測知電路展延了脈衝，並使黃色 LED 發亮，致您可以"看到"探測器"捕住"了一個脈衝。

您或許又可看到綠色與黃色 LED 發亮，而紅色 LED 不亮。這又是什麼意思？

代表邏輯 0 準位，以及一短暫邏輯 1 脈衝。

亦可能所有 LED 都發亮。這種情形就表示邏輯探測器的輸入，不斷地在邏輯 1 與邏輯 0 準位間迅速變化。

在後面一些實驗中，我們將以邏輯探測器檢查電路之輸

出，並且測知邏輯狀態與脈衝。這時候我們會說"……以探測器檢查……"或"……以邏輯探測器測量……"。這個意思就是教你將邏輯探測器連接到受測的電路上，以便能"看出"結果。

關掉計算機之電源。（爾後將簡稱關機）。

實驗2

設備位址解碼器之應用

目的

這個實驗使您能探知界面麵包板印刷電路板上之設備位址解碼電路的用法，這個解碼器由於在爾後所有的實驗中都會用得到，因此，您必須徹底了解其用法。

討論

這個實驗以A15～A0之位址位元辨認輸入/輸出設備所使用的特定位址。我們將以位址開關設定某一範圍內的位址，並以邏輯探測器檢查解碼器電路的動作。實驗亦會用到一個SN7402 NOR閘IC。

積體電路(IC)之接脚圖(圖6－3)

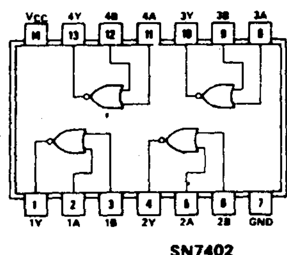


圖6－3 SN7402 NOR 閘之接脚圖

第1步

現在麵包板上應未接任何電路。若有電路，請將之除去。在這個實驗內，界面之解碼器部份會用到整個 16位元的位址巴士。記得，下半(LO)位址設定開關(IC－6)的最下面一個開關必須置於“M”或“ON”的位置。

第2步

將A15～A14 之所有位址位元的浮降開關置於邏輯1的位置。記得，切勿動了“M”開關。您知道SN74154 解碼器將解碼的是那些位址嗎？這個區間的位址有那些在ADDRESS輸出插座上有的呢？參考一下5－4圖。

區間內65520至65535之位址將爲4線對16線的解碼器(SN74154　)所解碼。由於解碼器僅提供了"下面"(較低)八個位址，因此，唯有65520至65527之位址可得。

第3步

打開計算機之電源。若原來您正在執行程式，則請按RESET鍵。以邏輯探測器測試ADDRESS插座的八個位址輸出。其中有一個解碼器輸出動作(有脈衝)嗎？由於現在根本不執行程式，因此，這是您所想像的嗎？

有兩個輸出應該動作，0與4，分別對應於位址65520與65524。計算機未在執行BASIC 程式時，其將執行一個一直監聽著鍵盤(看有無按鍵發生)的組合語言程式。因此，位址解碼電路應一直都在解碼位址。

第4步

將圖6－4所示的電路接好。記得，電源接脚，14 脚，一定要接至＋5伏特，而且接地接脚，7脚，要接至電源

接地。SN7402的接脚圖請參閱圖6－3。SN7402您亦可換成SN74LS02。此時，輸出A，B，及C暫不接至任何電路。

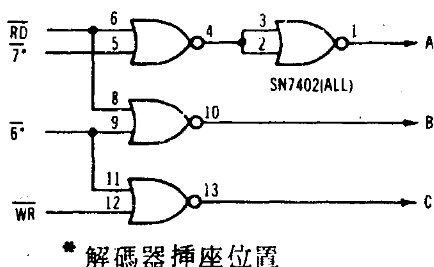


圖6－4　功能脈衝產生電路

第5步

將A15～A4位址位元的開關設定成位址49312，寫成二進制即11000000　10100000₂。記得，最低次四位元應忽略。當位址開關如此設定時，會產生那些位址呢？

49312至49327之位址將被解碼，但只有49312至49319之位址可用。

第6步

將下列程式打入計算機，並教計算機開始執行：

```
10  A = PEEK(49318)
20  GOTO 10
```

以邏輯探測器監聽著解碼器輸出，並記下您所看到的一切：

您應看到" 6 "號輸出動作，而且其它一些輸出亦可能動作。

接著，監聽邏輯閘輸出A，B，與C，並記下邏輯探測器所測知的結果：

	邏輯0	邏輯1	脈衝
A			
B			
C			

這就是您所預料的嗎？您能做個解釋嗎？

是的，這就是我們所預期的。因爲，程式中的輸入命令(

PEEK ）指明位址 49318 的設備作爲輸入設備，而且經解碼的位址在解碼器的“ 6 ”號輸出上。因此，唯有“ B ”輸出該動作。程式中未再指及任何其它輸入設備，亦未指及任何輸出設備。

第 7 步

將程式 10 號一列中所含之位址改成 49325 ，使 10號一列變成

```
10    A = PEEK ( 49325 )
```

然後再度執行程式且測試邏輯閘輸出 A ，B ，與 C 。請問有那些輸出動作，表示脈衝出現嗎？何故？

不應有任何輸出動作，因爲，設備位址 49325 並未裝在電路上。此外，位址 49325 在麵包板上亦沒有。在 49312 至 49327 的位址區間內，只有 49312 至 49319 幾個位址在 ADDRESS 插座上才有。

第 8 步

將 10 號一列再改成

```
10  A=PEEK(49318):B=PEEK(49319)
```

再執行程式時，脈衝出現於電路中之何處呢？

您應該發現輸出 A 與 B 動作。輸出 C 不動作，因爲，其爲一輸出控制脈衝，而程式中根本無輸出指令（ POKE ）。

第 9 步

再將程式修改。改變 10 號一列，使其能控制輸出設備 49318 。此時， 10 號一列的述句應如：

```
10  POKE 49318,0
```

資料值可使用 0 至 255（ 含 ）之間的任何值。執行程式，並測試輸出 A ， B ，及 C 。您想那一個輸出應動作呢？事實亦然嗎？

由於 POKE 指令爲輸出指令，且位址 49318 對應於解碼器之

“6”號輸出，故輸出C動作。看到B輸出亦動作您可能會覺得很驚訝！Apple之BASIC解釋程式在執行POKE指令時，計算機系統都會做一個“先讀取再寫入”（read-before-write）的動作。在設計界面電路的過程中記住這一點。

第10步

您能重新配組位址解碼部份之開關，使解碼器產生50944至50951之間的位址嗎？您如何做呢？這幾個位址眞的出來了嗎？

是的，可以改變開關設定，使解碼器產生這些位址。首先，將位址換成二進制：50944 = 11000111 00000000。接著，重新設定A15～A8以及A7～A4的開關。這時候，解碼器之“6”與“7”輸出所對應的位址各爲何呢？使用PEEK指令，以這個實驗一直在用的程式作個測試。您應可在邏輯閘之A與B輸出看到脈衝。

一旦測試過後，記得將位址開關還原成位址11000000 10100000_2 的狀態。

勿將電路拆掉，其還有用。不過，程式將不用了，因此，您可將麵包板與計算機的電源關掉。

實驗 3

使用設備選取脈衝

目的

在這個實驗您可看到以設備選取脈衝控制外部設備的用法。PEEK與POKE命令雖然通常均用以控制資訊流通，但事實上其亦可用以產生控制外部設備所用的脈衝。

討論

這個實驗以設備選取脈衝啓動或關閉一個簡單的設備。我們以邏輯探測器作“設備”，而且以兩個軟體產生的脈衝控制一個簡單的正反器。

IC接脚圖(圖 6 － 5)

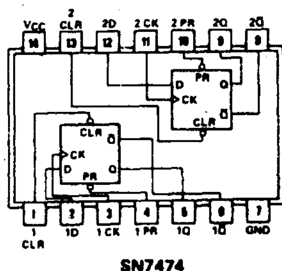


圖 6 － 5 SN7402與SN7474晶片之接脚圖

第 1 步

實驗 2 所使用的設備選取電路在這個實驗中亦用到。若還沒接，請您將之按圖 6 — 4 接好。

第 2 步

將 SN7474 正反器接成如圖 6 － 6 所示的樣子。圖中，SN7474 之“ D ”輸入處的“ 1 ”是代表這個輸入加邏輯 1（ + 5 伏特 ）。同樣地，“ 0 ”則代表接邏輯 0 或接地。以 0 與 1 代表只要是在強調其爲邏輯準位連接，而非帶電源連接。正反器之 Q 輸出應爲唯一連接至邏輯探測器的設備。SN7474 正反器記得要加電源； 14 脚接＋ 5 伏特，第 7 脚接地。

第 3 步

在電路內，WR 49318脈衝（ 信號 C ）將使正反器之輸出變爲邏輯 1 ，而 $\overline{\text{RD 49319}}$ 脈衝（ 信號 A ）將使正反器之輸出清除爲 0 。由於正反器平常都穩定於兩種狀態之其中一種狀態，因此，一旦受 RD 49319脈衝推動時，Q 輸出將保持於邏輯 1 狀態，而且這種狀態將一直維持至電源被拿掉或其被 WR 49318 脈衝清除成邏輯 0 爲止。

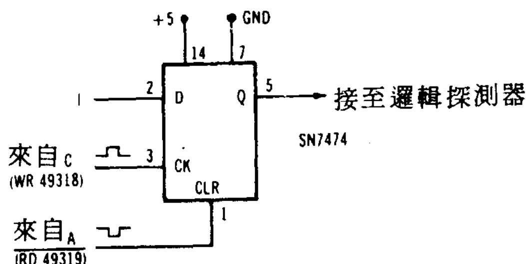


圖 6－6　簡單的正反器控制電路

將下列程式打入計算機，並加以執行：

```
10  A = PEEK(49319)
20  POKE 49318,0
30  FOR T = TO 300: NEXT T
40  A = PEEK(49319)
50  FOR T = 0 TO 300: NEXT T
60  GOTO 20
```

不管邏輯探測器脈衝之 LED 的閃爍。邏輯 0 與邏輯 1 LED 的結果各如何？

邏輯 1 與邏輯 0 LED 輪流地反復閃爍。

第 4 步

將 50 號一列之延時常式改成：

```
50  FOR T = 0 TO 1000: NEXT T
```

改完後，再執行程式。試問程式改變過後的結果如何？

邏輯 0 LED 亮更長一段時間。由此可見，我們可以產生間隔一段已知時間，譬如說 1 秒，的控制脈衝。

第 5 步

您能求出產生 1 秒鐘之週期時，FOR……；NEXT T 述句所必須產生的軟體延遲嗎？不斷地改變與測試各種延遲計數，直至週期相當接近 1 秒為止。或許您亦可先試 10 秒鐘之週期，然後再將之除 10 ，獲得 1 秒鐘之週期。最後您所獲得的延遲計數（亦即，述句中緊接 TO 之數值）為若干呢？我們發現，延遲述句

```
FOR T = 0 TO 780: NEXT T
```

執行起來大約 1 秒鐘。

第 6 步

現在，您可藉用 BASIC 的威力，告訴計算機，每一個 LED 應亮多久。打入且執行下面的程式。這個程式會先問您每一個 LED 欲亮多久（以秒計），然後才開始執行程式。

```
10  A = PEEK(49319)
20  INPUT "RED LED PERIOD ";Q
30  INPUT "  GREEN LED PERIOD "; R
40  PRINT "TOTAL CYCLE PERIOD "; Q+R; " SECONDS"
50  POKE 49318,0
60  FOR S = 1 TO Q
70  FOR T = 0 TO 780: NEXT T
80  NEXT S
90  A = PEEK(49319)
100 FOR S = 1 TO R
110 FOR T = 0 TO 780: NEXT T
120 NEXT S
130 GOTO 50
```

程式執行時，眞正的延遲時間可能稍爲加長一點。何故呢？

額外的程式步驟（FOR S=1 TO Q, FOR S=1 TO R 與 NEXT S ）增加了程式的全部執行時間，但由於幅度很小，所以還是看不太出來。

這個程式說明了什麼？

其說明了許多原理；以簡單的程式與簡單的電路控制外部設備。其亦說明了BASIC 以極簡單的程式步驟控制外部設備的威力。記住，雖然這樣，但BASIC 的速度還是相當慢。

雖然用了 PEEK 與 POKE 命令，但正反器界面的成功却未關係到任何資料或資訊的實際傳遞。正反器單獨受了設備選取脈衝的控制，或轉態。這個原理在需要控制信號或控制脈衝，但無資料傳遞時亦經常用到。

請務必記得，每當Apple計算機之BASIC 解釋程式在執行 POKE指令時，計算機一定**先做了一次讀取動作，然後再寫入**。因此，若您選擇以 POKE命令產生設備選取脈衝作控制，您則必須記住，**APPLE同時亦對同一位置做了一次讀取**。因此，若您同一位址使用了兩個控制脈衝，譬如說 WR XYZ 與RD XYZ ，則在以 POKE XYZ 指令作寫入時，RD XYZ 同樣亦會動作。

SN7474 正反器電路可以自麵包板拆掉了，但 SN7402 電路則必須留著。程式不再用了，因此，您可將系統的電源關掉。

實驗 4

構成輸入口

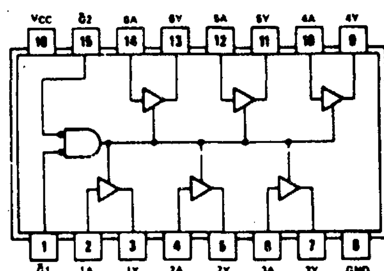
目的

此實驗的目的是在以三態緩衝器電路構成一個輸入口。

討論

這個實驗中您所組成的簡單八位元輸入口，主要在將資料輸入計算機。其它還有數個實驗將用到此一輸入口。前面用過的設備選取電路，這個實驗將再用到。實驗亦用到SN 74365或DM 8095三態緩衝器晶片。

IC接脚圖(圖 6－7)



SN74365A
SN74LS365

圖 6－7　SN74365或DM8095三態緩衝器晶片之接脚圖

第1步

實驗 2 所設計的閘控電路將用於這個實驗中。倘若您的麵包板上沒有這個電路，請參考圖 6 － 4 ，將之裝上。這時候，計算機與麵包板的電源應該是關掉的。

第 2 步

接好圖 6 － 8 所示的輸入口電路。總共需要兩個 SN 74365 (DM8095)三態緩衝器 IC。

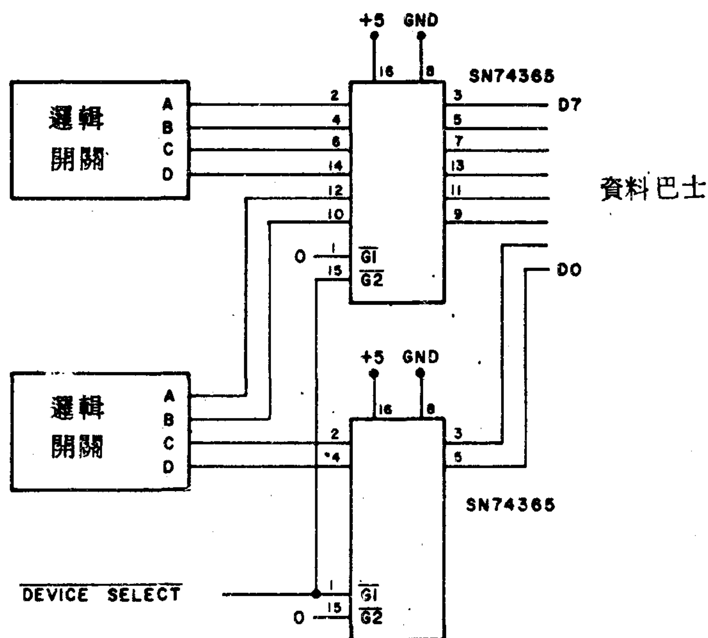


圖6－8 簡單的八位元輸入口

第3步

注意到，在這個電路，三態緩衝器晶片的兩個致能輸入僅用了一個。未用的輸入接地（或接邏輯0）了。由此可見，功能脈衝與設備位址並未以內部閘組合。致能信號在晶片內只是傳經邏輯閘，到達三態緩衝器電路罷了！

將 $\overline{\text{DEVICE SELECT}}$ 線接至圖6－4所示的A點（

SN7402 的第 1 脚）。這個信號就是 $\overline{\text{RD 49319}}$。

圖 6 － 8 中之"邏輯開關"即代表能在輸入口之八個個別輸入端產生邏輯 1 或邏輯 0 信號的開關。這些可以簡單的跳線接至電源巴士之+5 伏特與接地端代替。這種邏輯功能在附錄中亦可見。

第 4 步

一旦輸入口做好，且 SN7402 NOR 閘亦供應了設備選取脈衝之後，即打入且執行下列測試程式：

```
10 PRINT PEEK(49319): GOTO 10
```

在這個程式執行時，螢幕上顯示了些什麼？改變邏輯開關對顯示值有影響嗎？這是否正如您所想像的呢？

螢幕顯示了 255 ，相當於 11111111_2。改變邏輯開關對顯示數值無影響。起初，您一定想顯示數值會改成您後來所設定者。但事實却不然。何故呢？

界面電路並無輸入請求（ $\overline{\text{INP REQ}}$ ）信號，好將兩個巴士緩衝器置於輸入型態。

第5步

將SN7402的A，或 $\overline{\text{RD 49319}}$ ，信號與控制信號插座之INP REQ 部份的W輸入接在一起。這個信號將使8216巴士緩衝器動作於輸入型態。

接好之後，令程式重新開始，並改變開關設定。這回開關設定的改變會反映在顯示的數目上嗎？連續測驗幾次看看。

這回開關值就傳到計算機了，其同時被轉換成十進數，並顯示在螢幕上。

若您希望看到二進值，則您可打入且執行下面的程式。這個程式將不斷地顯示二進數。

```
10  A = 128
20  B = PEEK(49319)
30  FOR Q = 1 TO 8
40  IF B-A<0 THEN GOTO 100
50  PRINT "1";
60  B = B-A
65  A = A/2
70  NEXT Q
75  PRINT
80  GOTO 10
100 PRINT "0";
110 GOTO 65
```

若您要改變開關設定，然後獲得其二進等效，請將 10 號一列改成：

```
10 INPUT A$: HOME: A = 128
```

現在，只要隨時希望顯示輸入口之開關設定的二進值，您只需在Apple鍵盤上按個RETURN 鍵即可。當然，開關設定本身已成二進形式，因此，螢幕顯示之二進值與輸入口之每一位元間的關係，應極其明顯。

電路先勿拆掉，且電源亦留著勿關。這個程式與電路在下面一個實驗都還要再用。

實驗 5

多位元組輸入口

目的

這個實驗的目的乃在告訴您如何以BASIC 程式輸入與處理多位組的資訊。

討論

並非全部輸入設備送給Apple 計算機的資訊均僅一個位

元組。有些設備可能送出9個位元或更多。這個實驗將以實驗4所構成的輸入口模擬兩個輸入口。輸入口構成的細節請參考實驗4。作者建議您做完實驗4後再做這個實驗。

第1步

將實驗4所構成的輸入口連接至Apple計算機上。這個實驗必須使用這個電路。

第2步

在處理多位元組的資料時，程式必須設計成教Apple由最高次位元組至最低次位元組，依序處理資料之每一個位元組的方式。在這個實驗內，我們將以"M"代表最高次位元組（MSBY），並以"L"代表最低次位元組（LSBY）。因爲Apple會將八位元之數值均解釋成一介於0至255之間的十進數，因此，您能提出一個方程式或一系列運算，求得一雙位元組二進數的等效十進數嗎？

有。因爲低次位元組正好差一個256的因子。因此，我們可以下列方程式求得我們所要的十進值：

VALUE = (M * 256) + L

其中，VALUE 即爲十六位元資料字組的等效十進値。

第 3 步

爲了測試這個方程式，我們將下面程式打入計算機：

```
200  INPUT "SET MSBY ON SWITCHES ";A$
210  M = PEEK(49319)
220  INPUT "SET LSBY ON SWITCHES ";A$
230  L = PEEK (49319)
240  V = (256 * M) + L
250  PRINT V
260  GOTO 200
```

然後打入GOTO 200，並按RETURN 鍵，敎計算機開始執行程式。當計算機問您"SET MSBY ON SWITCHES ?"（高次位元組已設定在開關上了嗎？）時，即將十六位元資料的高次八位元設定在八個開關上。然後按 RETURN 鍵。稍後，當計算機又問您"SET LSBY ON SWITCHES ?"（低次位元組已設定在開關上了嗎？）時，即將資料低次八位元再設定於開關上。開關一設定好後，就按RETURN 鍵，讓計算機知道您已設定好了。此時，螢幕上就應顯示出十六位元資料値的等效十進値。分別試試下面所列的典型十六位元値，並記下結果，塡入第三欄內。結果對不對，您應能很快地就檢查出來（必要時可藉用手上型計算器）。

MSBY	LSBY	顯示十進值
11001010	11000001	
11000111	00011101	
00000001	10000001	

正確答案應依序爲51905，50973，以及385。

第4步

下面的程式乃二進輸出程式與雙位元組十進值計算程式的組合。使用這個程式，您可輸入兩個代表一十六位元值的位元組，計算機螢幕將顯示出此一十六位元值的二進形式與十進形式。

```
10  A = 32768
20  FOR S = 1 TO 2
30  FOR Q = 1 TO 8
40  IF B-A<0 THEN GOTO 100
50  PRINT "1";
60  B = B-A
65  A = A/2
70  NEXT Q
75  PRINT "    ";:NEXT S
80  PRINT: GOTO 200
100 PRINT "0";
110 GOTO 65
200 INPUT "SET MSBY ON SWITCHES "; A$
210 M = PEEK(49319)
220 INPUT "SET LSBY ON SWITCHES "; A$
230 L = PEEK(49319)
240 V = (256 * M) + L
250 HOME: PRINT V
260 B = V: GOTO 10
```

第 5 步

程式打完後，打入GOTO 200，然後按RETURN 鍵，教計算機執行程式。同樣地，將資料值之高次位元組與低次位元組分別設定在開關上。開關設定與螢幕所顯示之二進值之間應有關係存在。您應能輕易將二進值轉換成十進值，以檢查結果是否正確。為了能讓您易於與開關設定作個比較，螢幕上所顯示的二進值特別"分成"了兩段。

至此，您已知道Apple能如何將兩個位元組重新組合成一個十六位元的數值。您亦應能體會到其它型式的運算怎麼做。這個實驗雖然僅用了一個輸入口，但以另一個位址構成一個新的輸入口，以產生十六位元值的另一個位元組應是不難的事！

您或許也發現這個實驗以及上一個實驗使用了一個新的變數A $。這個變數是一個為了使程式能停在預定的點上，好讓實驗條件改變後計算機再繼續下去的"虛擬"變數。A $ 變數為一字串變數，每次按RETURN 鍵後，這個變數的值即被令為空字串——什麼都沒有。這正是一個叫停計算機，讓其在我們按完RETURN 鍵之後再開始的"技巧"。

這個實驗所用的界面電路在下一個實驗會在用到，因此，將之留著。軟體不用了，所以，計算機與界面的電源可以關掉。

實驗6

輸入口應用

目的

本實驗的目的乃在告訴您如何以輸入口作控制。

討論

在這個實驗內，八位元的輸入口將資料送給Apple，但Apple 却將資料當成非數值資料處理。依此，計算機即可監聽到八個外部設備的狀態。

第1步

若輸入口未接在Apple計算機上，請參考實驗4將之接上。底下的步驟要用到這個輸入口。

第2步

經常，計算機亦要處理告訴計算機有關外部設備之狀態的非數值資訊。根據這種方式，我們即可輕易知道那一個設

備現在正動作或不動作、閘門是打開或關閉、升降機正往上或往下等等。

將下面程式打入您的計算機，並敎計算機開始執行。這個程式說明了如何以一個數值敎計算機採取一些事先計劃好的動作：

```
10  INPUT A$: HOME
20  A = PEEK(49319)
30  IF A>127 THEN GOTO 70
40  PRINT "INPUT <= 127"
50  GOTO 10
70  PRINT "INPUT > 127"
80  GOTO 10
```

第 3 步

爲了讓計算機執行輸入與比較的步驟，您必須按 RETURN 鍵。在輸入口的邏輯開關上設定一個小於 127（00000000 至 01111110）的數值，然後按 RETURN 鍵。結果如何呢？以 127 或更大的數值（01111111 至 1111111）再試試看，結果又如何呢？當二進值等於 127（0111 1111）時，結果怎麼呢？每一個數值輸入計算機時，您都應看到正確的訊息。這個程式舉例說明了計算機如何根據一個數值作決策，有時候，單獨的位元值亦可作爲決策的基礎。實驗 4 之二進轉換程式使您能看到一十進值的二進等效。這個程式即根據個別的位元值作決策，致其可知道是否在每一位元位置上顯示一個" 1 "。

第4步

這個步驟亦使用基本的二進顯示常式，但不是顯示0與1，而是在邏輯1時顯示" ON "，在邏輯0時顯示"OFF"。你應自己有辦法更改實驗4之程式中的PRINT 述句，使其變成這種功能。不過，爲了方便起見，我們還是將程式列在下面。注意到，這個程式的列號碼已異於實驗4的程式（提高了）。若您一直未關機，則在打入這個新程式前，記得將原來舊有的程式除掉。除去舊有的程式可使用 NEW 命令。您只要打入 NEW ，然後按RETURN即可。

```
410  INPUT A$: HOME: A = 128
420  B = PEEK (49319)
430  FOR Q = 1 TO 8
440  IF B-A <0 THEN GOTO 500
450  PRINT "ON  ";
460  B = B-A
470  A = A/2
480  NEXT Q
490  GOTO 410
500  PRINT "OFF "
510  GOTO 470
```

注意：ON之後有兩個空格，而OFF之後有一個空格。這主要在產生相等的空間。

執行這個程式。記得，開關要設定，而且之後必須按RETURN 鍵，計算機才有辦法做"轉換"與顯示。螢幕上應出現一排ON與OFF——邏輯1位元處出現 ON，邏輯0位元處出現OFF。願意的話，您亦可將ON與OFF 分別改成OPEN與CLOSED，或UP與DOWN ，或其它類似符號

等等。

第 5 步

第 4 步之簡易程式雖然有點用，但若能將ON與OFF訊息顯示成整行（縱的方向）則更佳。產生這種垂直顯示效果可應用 BASIC中的HTAB與VTAB兩個命令。以下所示即爲經修改過的程式，修改過的列上分別打有星號（ * ）。450 與 500 號列上的ON與OFF之後必須分別留空格。

```
*400  H = 20: V = 8
 410  INPUT A$: HOME : A = 128
 420  B = PEEK(49319)
 430  FOR Q = 1 TO 8
 440  IF B-A<0 THEN GOTO 500
*450  HTAB H: VTAB V: PRINT "ON ";
 460  B = B-A
*470  A = A/2: V = V+1
 480  NEXT Q
*490  GOTO 400
*500  HTAB H: VTAB V: PRINT "OFF ";
 510  GOTO 470
```

由於HTAB與VTAB 命令令指標垂直地移動，因此，此時您所看到的ON與OFF 應是成行的。

由此可見，ON與OFF 狀況可顯示成多種方式。事實上，有些計算機上之圖形符號與文數字可混合在一塊兒，以使ON/OFF 狀況顯示成近乎像設備的圖樣。

在程式執行過程中，試著改變開關設定狀態，以確實證實程式與輸入口均能正確動作。

第6步

或許您希望能連續地執行程式，以便能改變開關設定狀態以及ON/OFF 狀況，而且每次欲作新顯示時不必按RETURN鍵。前面說過，INPUT A$是一個令計算機停下來，一直等到您再按RETURN 鍵才繼續動作的"虛擬"輸入指令。因此，我們現在拿掉這個述句，使410號一列變成:

```
410  HOME: A = 128
```

然後再執行程式。這下子計算機能作合理的顯示嗎？何故？

由於每次計算機重新執行程式時，HOME指令均將整個螢幕完全清除，並將指標(cursor)定位於最左上角落上。所以，螢幕顯示很糟糕地閃爍不定。這不僅費時，而且減慢了顯示的速度。您能提議一種辦法，進一步修改程式，以減少或甚至免除這種閃爍現象嗎？

第 7 步

除去HOME指令，即可減短Apple清除整個螢幕並且令指標回至螢幕之左上角落所需的時間。使用HTAB與VTAB指令時，這兩個指令能將指標定位於正確的位置上，以印出每一列上的ON或OFF，每一位元一個。不過，若450號一列上之“ ON ”後面不留空格，則ON印出時將無法蓋掉OFF之最後一個F ，因此您會看到印出結果爲 ONF ，而非ON。由此可見，爲了除去餘留在每一列上的文字，您必須適當地使用空格。

因此，我們建議您，將程式410號一列改成：

```
410 A = 128
```

然後，打入HOME : GOTO 400 ，再按ENTER ，教計算機重新執行程式。若您不使用HOME命令，則計算機將直接在舊有的螢幕顯示上寫上新的資料。加上這個HOME命令，在程式開始前，計算機就會先清除整個螢幕。

第 8 步

我們亦可以VTAB與HTAB 命令在八列顯示訊息之每一列前印出標題或說明。以下即爲數種標題，您可隨意增加或修改：

```
 5  HOME
10  VTAB 8: HTAB 1
15  PRINT "ACID PUMP";
20  VTAB 9: HTAB 1
25  PRINT "BASE PUMP";
30  VTAB 10: HTAB 1
35  PRINT "HEATER";
40  VTAB 11: HTAB 1
45  PRINT "MIXER";
50  VTAB 12: HTAB 1
55  PRINT  "FLUSH CYCLE";
60  VTAB 13: HTAB 1
65  PRINT "DISHWASHER";
70  VTAB 14: HTAB 1
75  PRINT "VACUUM":
80  VTAB 15: HTAB 1
85  PRINT "DRYER";
```

若您計劃繼續做實驗7，我們建議您將這幾列加入程式內。加入這幾列後，記得要再將程式重新測試一次。

這個實驗所用的軟體與硬體在下一個實驗都會用到，所以，請勿將電路拆掉或將電源拔掉。

實驗7

輸入口應用(之二)

目的

本實驗的目的乃在教您如何對資料作邏輯運算。

討論

這個實驗將使用 AND 運算，加且對八個外部"察覺器"所產生的ON/OFF 訊息作運算。實驗將以這些察覺器的狀況觸發計算機的動作。

第1步

這個實驗所使用的程式與實驗 6 所使用者相同。若這個程式尚未完全打入您的計算機，則請您立即打入，打完後並加以測試一遍。若在前一個實驗時已打完而且測試過了，則請您再將之與以下之列表核對比較一下：

```
5   HOME
10  VTAB 8: HTAB 1
15  PRINT "ACID PUMP";
20  VTAB 9: HTAB 1
25  PRINT "BASE PUMP";
30  VTAB 10:  HTAB 1
35  PRINT "HEATER";
40  VTAB 11:  HTAB 1
45  PRINT "MIXER";
50  VTAB 12:  HTAB 1
55  PRINT "FLUSH CYCLE";
60  VTAB 13: HTAB 1
65  PRINT "DISHWASHER";
7(  VTAB 14: HTAB 1
75  PRINT "VACUUM";
80  VTAB 15:  HTAB 1
85  PRINT "DRYER";
400 H = 20:  V = 8
```

```
410  A = 128
420  B = PEEK(49319)
430  FOR Q = 1 TO 8
440  IF B-A<0 THEN GOTO 500
450  HTAB H: VTAB V: PRINT "ON ";
460  B = B-A
470  A = A/2: V = V+1
480  NEXT Q
490  GOTO 400
500  HTAB H: VTAB V: PRINT "OFF ";
510  GOTO 470
```

在正確的取入與測試過後，程式應產生一如表6－1所示的顯示。其中，因您在輸入口的邏輯開關設定不同而異，計算機所顯示之ON與OFF的狀況或許有所出入。

表6－1　控制程式的輸出

ACID PUMP	ON
BASE PUMP	OFF
HEATER	ON
MIXER	ON
FLUSH CYCLE	ON
DISHWASHER	ON
VACUUM	OFF
DRYER	OFF

第2步

沿表6－1之邊緣，記下每一標題對應於輸入口的那一個位元。這可經由測試輸入位元或分析程式達成。您應發現，“ACID PUMP”即爲D7位元，“BASE PUMP”爲D6位元，……“DRYER”爲D0位元等等。

第 3 步

參考第 4 章之例題 4 － 3 ，並以 Apple 之監督程式將這個組合語言程式打入計算機。您只要打 CALL- 151 ，然後按 RETURN 鍵即可進入監督程式。檢查看程式都打對了沒。記得，監督程式使用十六進數。若您不知道如何使用監督程式，請逕自參考" Apple Ⅱ 參考手册"，或遵循下列步驟:

1. 按 RESET 鍵，並打入 CALL－151 ，然後按 RETURN 鍵。這時 Apple 應回以星號（ * ）。
2. 打入 0300 ： 00 00 00 48 AD 00 03 2D 01 03 8D 02 03 68 60 。每一組（兩位數字）數目之間均留一個空格。以 00 作程式的最初三個值。
3. 按 RETURN 鍵，打入 02 FF ，按 RETURN 鍵，再按 RETURN 鍵兩次，然後將資料與例 4 － 3 所列者作個檢查比較。

第 4 步

打入且執行以下所示的程式，以測試組合語言程式。為了檢查結果是否正確，您可以紙和筆做一些必要的十進至二進以及二進至十進轉換。按 RESET 回到 BASIC 。

```
1000  POKE 10,76:POKE 11,03:POKE 12,03
1010  INPUT "MASK BYTE "; M: POKE 768,M
1020  INPUT "DATA BYTE "; D: POKE 769,D
1030  Q = USR(0): PRINT "ANSWER "; PEEK(770)
1040  GOTO 1010
```

若所得答案與您手算者相同，則繼續進行下一個步驟。否則，仔細的檢查一下每一個組合語言程式步驟，然後再重新測試程式一次。記得，錯誤亦可能發生在您的手算上。

第5步

現在修改您的程式，使其能測知何時DISHWASHER（洗碟機），DRYER（烘乾機），或VACUUM（吸塵器）三件電器中有任一者打開，以及何時ACID PUMP 與 BASE PUMP兩者同時打開。雖然可能還有其它許多種解法，但這時我們可以使用邏輯 AND 組合語言副程式。

您能提議一種作這些決定的方法嗎？作者建議您再複習一下第4章所提過的邏輯 AND 運算。如表6－2所示地考慮這些運算。

表 6 — 2　欲測知之控制狀況

D7	D6	D5	D4	D3	D2	D1	D0	
1	1	X	X	X	X	X	X	ACID 與 BASE PUMPS 兩者均 ON
X	X	X	X	X	0	0	1	任一件電器品 ON
X	X	X	X	X	0	1	0	
X	X	X	X	X	0	1	1	
X	X	X	X	X	1	0	0	
X	X	X	X	X	1	0	1	
X	X	X	X	X	1	1	0	
X	X	X	X	X	1	1	1	

X = 無關，邏輯 0 或邏輯 1 均可。

第 6 步

我們可以邏輯 AND 運算遮掉不必要的位元，抽氣機（PUMP）測試時即遮掉 D5 ～ D0，電器測試時遮掉 D7 ～ D3。因此，總共需建立兩個"面罩"，抽氣機一個，電器品一個。這些面罩為何呢？

抽氣機測試面罩為 11000000_2（＝ 192），而電器品面罩為 $0000\ 0111_2$（＝ 7 ）。以這些面罩與察覺器或邏輯開關之輸入值作 AND ，我們所想要的位元即可經面罩"濾出"。

第 7 步

現在，兩個面罩都弄好了，擬議一些可用以求知這些"

過濾"出之位元值的程式步驟吧!您必須考慮到每一個個別位元以及這些位元的十進等效。若需要，您可使用新的變數。

我們使用一個新變數C代表察覺器所輸入的數值。這樣子變數B就可以獨立用於程式的ON/OFF 顯示部份。若您使用變數B，則您會發覺其恒爲零。讓你自己試著找出爲什麼。我們使用。

```
POKE 768,7:POKE 769:C:Q=USR(0)
IF PEEK(770) = 0 THEN . . .
```

或

```
POKE 768,7:POKE 769,C:Q=USR(0)
IF PEEK(770) >0 THEN . . .
```

測知電器品。並以類似的步驟測知抽氣機。在任一種情況，某一種狀況時，計算機均執行THEN……述句，相反狀況時，計算機則繼續執行下一個緊接述句。

第8步

為了測試您的程式觀念，在基本的旗號測知程式內再增加步驟，使程式在兩個PUMP都同時打開時，能在螢幕上印出DANGER（危險），且在任一種電器打開時印出

APPLIANCE（電器品）。將您所設計的程式步驟寫在下面空白內，並仔細檢討後，再改變程式。記得，若您想藉用組合語言副程式，您將需要一像第 4 步之 1000 號一列的述句。這個程式列設定 USR 命令所用到之三個位置的初值，使其能將計算機引導至正確副程式的開始。

您的程式看起來或許像：

```
420  B = PEEK(49319): C = B
 .
 .
 .
490  GOTO 600
 .
 .
 .
600  POKE 768,7: POKE 769,C
605  Q = USR(0)
610  IF PEEK(770) = 0 THEN 700
615  HTAB 20:VTAB 17: PRINT "APPLIANCES";
620  POKE 768,192
625  Q = USR(0)
630  IF PEEK(770) <> 192 THEN 800
635  HTAB 20:VTAB 18: PRINT "DANGER";
640  GOTO 400
700  HTAB 20:VTAB 17: PRINT "           ";
710  GOTO 620
800  HTAB 20:VTAB 18: PRINT "           ".
810  GOTO 400
```

測試一下您的程式。您或許會忘了自螢幕清除掉APPLIANCES與DANGER顯示的程式步驟。或許您也忘了以三個POKE指令將 USR 命令所需的資訊取入適當的記憶位置。這時，您只要打入POKE命令，緊接按RETURN即可，不必再增加其它任何程式步驟。唯一所需的就是再執行程式一次。

700與800號印出空白的命令就是在清除螢幕上所顯示的APPLIANCES與DANGER 信號。這個程式可再寫得更複雜一點，以包含使用黑白顛倒顯示，或在程式測及緊急狀況時使用閃爍顯示的步驟。至此您應了解軟體既可處理數學運算，亦可處理邏輯運算了。您也該了解組合語言副程式的使用並不難。

雖然組合語言 AND 運算程式下面還要再用，但您可暫

先關掉計算機的電源。由於輸入口電路還要再用，因此，暫且先慢一點拆除。

實驗 8

構成輸出口

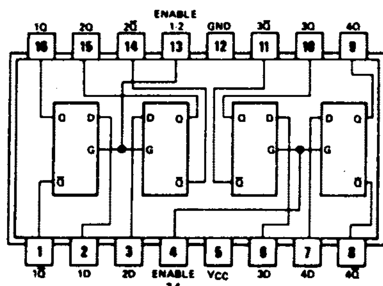
目的

本實驗的目的乃在讓您構成一個簡單的八位元輸出口，並探討其用途。

討論

這個實驗以一個簡單的八位元鎖住器電路組成一個輸出口。這個輸出口將用於目前這個實驗，以及爾後幾些必須將資訊送給外部設備的實驗中。實驗將使用兩個 SN7475 四重鎖住器 IC。

IC接脚圖(圖 6 — 9)



功能表
(每一鎖住器)

INPUTS		OUTPUTS	
D	G	Q	$\bar{Q}$
L	H	L	H
H	H	H	L
X	L	Q_0	$\bar{Q}_0$

H = high level, L = low level, X = irrelevant
Q_0 = the level of Q before the high-to-low transition of G

圖 6 — 9　SN7475四位元鎖住器晶片之接脚圖

第 1 步

這個實驗將用到實驗 2 所用過之閘控電路。若您的免焊麵包板上沒有這個電路，我們建議再做一次實驗 2 ，然後再做這個實驗。若不願這樣做，您亦可直接接上然後使用這個閘控電路。電路詳圖請參考圖 6 — 4 。

第 2 步

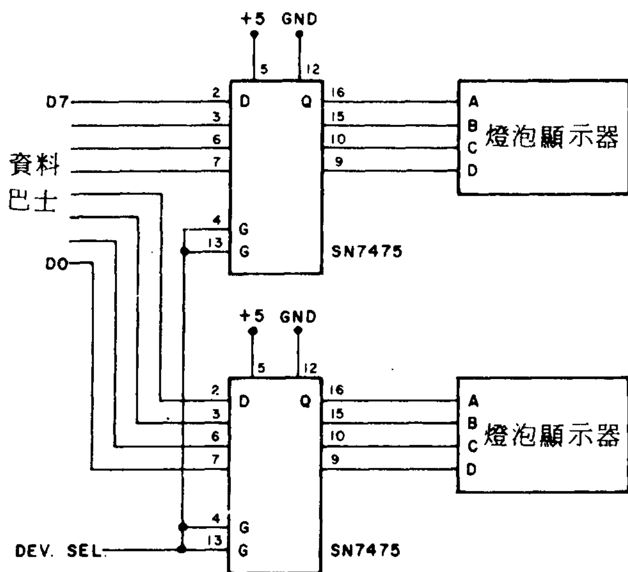


圖 6 —10　簡易的八位元輸出口電路圖

接好如圖 6 － 10 所示的電路。 這個電路總共需要兩個 SN7475 鎖住器 IC ，以及八個個別之燈泡顯示器或相等之邏輯準位測知電路。設備選取輸入 DEV SEL此時暫時不接。

第 3 步

參考圖 6 － 4 所示的電路。試著決定以 A ，B ，與 C 三個控制輸出中之那一個輸出，控制連接至 DEV SEL 線之鎖住器致能輸入。您選用那一個呢？爲什麼？



由於A輸出，$\overline{\text{RD 49319}}$，已經用了，而且RD 49318僅解碼作輸入口，不能用。因此，只有選用C輸出── WR 49318。這個輸出產生一個與SN7475鎖住器晶片所需者同一類的正脈衝，而且亦解碼作輸出用。你應還記得，印刷電路板上之解碼器的 7 與 6 輸出接脚，實際上分別對應於解碼位址 49319 與 49318。

將 SN7402 的第 13 接脚與兩個 SN7475 鎖住器晶片的第 4 及第 13 接脚接在一起。這就是圖 6 － 10 所示的DEV SEL 接點。

第 4 步

為了測試輸出口，將下列程式打入計算機：

```
10  A = 0
20  POKE 49318,A
30  END
```

如程式所示的，預先將變數A令為零，然後再執行程式。燈泡顯示器有何變化？

因爲計算機送零到輸出口，因此，指示燈應全不亮。此時，將A令爲255，再執行程式一次看看。所有 LED 都應發亮。若不是這種情況，則檢查一下您的電路與測試程式。

第5步

程式可改成使您能輕易自鍵盤打入新值。這個新的程式乃爲：

```
10  INPUT A
20  POKE  49318,A
30  GOTO 10
```

雖然您可測試您所選定的任何值，但我們建議您使用 2 的乘冪：0，1，2，4，8 等等。因爲，這樣可以測到每一個個別的LED。

一個八位元輸出口僅能顯示介於0 至 255 之間的數值，當您試圖輸出一個超出此一範圍的數值時，結果會如何呢？您想您會看到"一部份"（如低次的八位元）值嗎？以256 試著執行程式，看看，結果會如何？

```
?ILLEGAL QUANTITY ERROR IN 20
```

代表值不在函數所要求的正確範圍內。錯誤訊息中同時列出了錯誤所在的列號碼。試試看，負數亦會得到同樣的結果。

第6步

令程式重新開始，並打入一 90 之數值。此時，燈泡指示器應顯示0101 1010。再試著打入－24看看。當計算機發現錯誤且顯示出錯誤訊息時，螢幕所顯示的數值是否改變呢？

不。錯誤狀況均在POKE功能之使用前發現。您想Apple會如何處理小數呢？打入一十進小數，譬如6.01，顯示如何？

Apple將"截去"數目之小數部份。再試試其它的數目看看。

第7步

您能設計一個將一個數值由0每次加1，一直加至255，並將每一個新值顯示在 LED 上的簡短程式嗎？將這個程

式寫在底下的空白內，然後加以測試。您看到什麼結果？您能讓程式自動廻路（ loop ），一直反復地累加再累加嗎？

我們使用下面的程式：

```
10  FOR A = 0 TO 255
20  POKE 49318,A
30  NEXT A
40  GOTO 10
```

記得，若不造成錯誤的話，數值無法大於 255 或小於 0 。為了不讓 LED 亮滅地閃爍得太快，我們可在程式內加入一段短暫的時間延遲。這個延時步驟可如：

```
25  FOR T = 0 TO 500: NEXT T
```

至此，您應該發現，輸出口之構成以及軟體控制均十分容易。

這個輸出口在下面的實驗會用到，但您可以把電源先關掉。

實驗 9

輸出口與輸入口交相作用

目的

這個實驗的目的乃在告訴您，輸入口指令與輸出口指令可如何用在同一個程式內。

討論

在界面電路內，輸入口與輸出口經常會一起使用。這些輸入/輸出口受同一個程式之PEEK與POKE命令的控制，而且口與口之間經常有資訊的傳遞，這個實驗告訴您，這些口如何一起用於一個簡單的電路內。

第1步

前面使用過的簡單輸入口(實驗7)與輸出口(實驗8)在這個實驗內將繼續用到。有關的電路細節亦要參考實驗2，3與8。

第2步

輸入口與輸出口旣接好了以後，即將下列程式打入您的計算機，並教計算機開始執行。這個程式用以測試輸入/輸出口電路。

```
10  A = PEEK(49319)
20  POKE 49318,A
30  GOTO 10
```

當您設定輸入口的邏輯開關時，您會看到，對應的輸出口位元亦會隨著開關動作改變。萬一沒有，請再檢查一下電路及程式。

第3步

這個步驟將自鍵盤打入兩個值，然後顯示於 LED 上。目前，您應該自己會寫這麼樣一個程式了。試試看：

記得我們曾經用過下述模擬一高次位元組（MSBY ）與一低次位元組（LSBY）的程式：

```
10  INPUT "MSBY ";A$: M = PEEK(49319)
20  INPUT "LSBY ";A$: L = PEEK(49319)
30  POKE 49318,M
40  INPUT A$
50  POKE 49318,L
60  GOTO 10
```

這個程式以字串變數A $ 作" 虛擬" 變數" 叫停" 計算機，使您能做些必要的動作後，再讓程式繼續執行。

第 4 步

令計算機執行您的程式。您應自己會打入這兩個值。當您打入 RUN　RETURN 時，計算機即準備好接受您自開關輸入MSBY。MSBY 值設定好後，即按 RETURN鍵，讓計算機能執行獲取資料的步驟。然後，再於開關上設定 LSBY 的值，並再度按 RETURN鍵。一獲得 LSBY 時，程式即會將MSBY顯示出。按 RETURN 鍵，您即可教計算機顯示出 LSBY。

第 5 步

這個程式說明了計算機如何獲取並存起稍後欲顯示的資料。八位元的資訊很容易做。請問，介於 0 至 65535 間的數目如何以兩個輸出口顯示出呢？

這些數目必須" 分割"成一八位元的MSBY與一八位元的LSBY。您能說說這怎麼做嗎？

將數目除以 256，所得結果之整數部份即爲MSBY。譬如，若數目爲 10923，則我們首先將

10923/256 = 42.668

所得結果之整數部份 42，經轉換成一八位元二進數後，即爲MSBY之值。求出MSBY後，低次位元組LSBY 可以下面方法求得：

10923 − (42 * 256) = 171

這時，171 轉換成八位元二進等效後，所得即爲我們所要的LSBY。

在Apple上，十進值轉換成八位元的二進等效可寫一個BASIC程式來做，您會寫嗎？

第 6 步

我們以下列的程式來做這個" 轉換"：

```
10  INPUT "VALUE "; V
20  M = V/256
30  L = V - INT(M) * 256
40  PRINT INT(M), L
50  INPUT A$
60  POKE 49318,M
70  INRUT A$
80  POKE 49318,L
90  GOTO 10
```

螢幕上會顯示MSBY與LSBY 的十進形式。爲了清楚起見，程式特以 INT 指令除去 M值之小數部份。若爲 POKE 運算，這就不需要了，因爲，十進小數部份會自動忽略。

第7步

將這個（或您自己寫的）程式打入計算機，並加以測試。欲令MSBY顯示在LED上時，您必須按RETURN鍵一次，且欲顯示LSBY時必須再按一次。

您可以打入大於65535的數值嗎？程式能不能轉換且顯示這種數值呢？

是的，可以打入，而且可以轉換，但却不能顯示。因爲，其所產生之MSBY值大於256。這等於一種錯誤狀況，您能設法防止嗎？

您可在程式內加入幾個程式步驟，在轉換之前先檢查值的範圍，同時亦可加入除去數目之小數部份的步驟。這些步驟可爲：

```
12  IF V < = 65535 AND V > = 0 THEN 18
14  PRINT "VALUE OUT OF RANGE, TRY AGAIN": GOTO 10
18  V = INT(V)
```

您可試著將這些步驟加入程式試試看。諸如此類的步驟使程式不致發生錯誤，且使程式取向於用者。當您在設計更複雜的程式時，最好隨時謹記這點。

實驗10

資料取入與顯示

目的

本實驗的目的乃在告訴您如何以輸入口獲得資訊，以及計算機如何將這些資訊存起，以便後來顯示於 LED 上。

討論

這個實驗自一三態的輸入口獲得 10 個資料值，且稍後自 LED 顯示出。實驗後面亦討論更富彈性的顯示觀念以及獲得更大量資料的方式。

第 1 步

前面介紹過的輸入口與輸出口在這個實驗中繼續使用。目前，您應已了解這些輸入與輸出口。若必要，您可再回頭參考實驗 2 ， 3 與 8 。萬一您尚未做過這些實驗，我們建議您還是先回過頭去做一做再來。

第 2 步

這個實驗，您以計算機獲得且顯示一組自輸入口所獲得的資料值。這些資料值可以諸如以下的軟體步驟獲得：

```
50  INPUT A$
60  Q = PEEK(49319)
70  INPUT A$
80  R = PEEK(49319)
```

這種方法以大量的軟體步驟獲得一小量的資訊。您能提供另一種較佳的辦法嗎？

我們可以一個廻路獲得一系列的資料，並且以一陣列 (array) 儲存這些資訊，而不必每來一個新的資料值即需將之令至一新的變數。請問您能設計一個可用以獲取十個資料值的簡短程式嗎？

這個程式可如：（注意到以陣列儲存資訊的用法）

```
 10  DIM A(10)
 20  PRINT "START"
 30  FOR P = 1 TO 10
 40  INPUT A$
 50  A(P) = PEEK(49319)
 60  NEXT P
 70  PRINT "START DISPLAY. . ."
 80  FOR P = 1 TO 10
 90  GET A$
100  PRINT A(P): POKE 49318,A(P)
110  NEXT P
120  PRINT "END OF RUN": END
```

執行這個程式時，每當欲獲得一個資料值，您即必須按 RETURN 鍵一次。當計算機在螢幕上印出" START DISPLAY……"時，每次您按一個RETURN鍵，計算機螢幕即會顯示出一個其已儲存的值。這個值同時亦會以二進形式顯示在LED上。注意，我們所用的是GET A$ 述句，而非 INPUT A$ 述句。這兩者有任何差別嗎？

是的，GET A$ 指令使螢幕不致顯示出問號（？），且任何文字鍵（A，&，1，等等）均可用以代替RETURN鍵。文數字符號在螢幕上不顯示出。其僅"清除"整個資料值顯示。

第3步

教計算機執行您自己所設計的程式，或我們給您的程式，以獲得十個資料值。一旦獲得這些資料值後，即以計算機將之顯示出。請問您看到什麼結果呢？

您應發現，所有資料值均正確儲存於記憶器，顯示於 LED

上，且印出於螢幕上。假設不願意讓這些資料值顯示於輸出口上，請問您能修改程式，使其僅將資料值顯示於螢幕上嗎?

是的，只要將 100 號一列改成

```
100   PRINT A(P)
```

並除去 90 號一列即可。

第 4 步

您亦可以 Apple 之低解像度圖形作業型態，將數值顯示成圖案形式。作者建議您以 HLIN 指令畫出一組代表自輸入口輸入之相對值的水平線。記得，使用 HLIN 指令時，螢幕面積之大小有一定的限度。每一方向至多僅能畫出 39 點。

將您所設計之顯示程式步驟寫在下列空白內：

產生水平棒形圖的程式步驟可爲：

```
 80  GR: COLOR = 5
 90  FOR P = 1 TO 10
100  D = A(P)/6.5
110  HLIN 0,D AT P
120  NEXT P
130  END
```

這幾個程式步驟可加在第 2 步所設計的程式內。試試這個程式或您自己所設計的程式看看。

這個程式特別將資料值除以 6.5，以使其值變成由 0 至39之間，而非原有之 0 至 255 之間。同時，每一條水平線之起始位置亦隨陣列之註標值逐次增進。A (1)時，資料自螢幕之頂端開始，以後之資料值則逐次往螢幕之下緣遞降。您亦可以 P 之值改變每一條水平線的色彩。

第 5 步

程式還可做其它的改變，將 INPUT A$ 述句換成一個延時迴路。這樣的意思等於每隔一段固定的時間才獲得一個資料值，時間間隔由延時迴路決定。同時，每次獲得資料值時亦不必再按 RETURN 鍵了。

改變程式，將 40 號一列之 INPUT A$ 述句換成一延時迴路，並使延時時間長至 2 或 3 秒左右。譬如，下面就是一個例子：

```
40  FOR T = 0 to 2000: NEXT T
```

將邏輯探測器連接至SN7402 NOR閘上之" A "輸出——第1脚。如此，三態輸入口每獲得一個資料值，邏輯探測器即會使紅色 LED 閃亮一次。由此您可知道，程式已獲得一個值了。

您或許希望將程式改成能獲得 10 個值以上。使用這個簡單的顯示常式，您最高可獲取39個值。

將程式作必要修改，以延時令輸入口之資料獲得取得同步。執行程式。您亦可再將延遲加長至您可輕易改變開關設定爲止。

這時，您的程式看起來應像：

```
10  DIM A(10)
20  PRINT "START"
30  FOR P = 1 TO 10
40  FOR T = 0 TO 2000: NEXT T
50  A(P) = PEEK(49319)
60  NEXT P
70  PRINT "START DISPLAY. . ."
80  GR: COLOR = 5
90  FOR P = 1 TO 10
100 D = A(P)/6.5
110 HLIN 0,D AT P
120 NEXT P
130 END
```

您是否發現，並非所有的輸入資料值均使顯示發生改變。試著打入0，1，2，3，……等等一直到9看看。您或許可將延遲再變慢，或將40號一列換回 INPUT A$ ，致您有足夠的時間變換開關之值。在您打入這些值時，顯示如何呢？何故？

0 至 6 在螢幕上顯示出同樣的值，而且 7 至 9 亦顯示同樣的值，但却比前面的值，0 至 6，大一個“方格”。這個原因是，所有的值都“縮短”在 0 至 39 之間，致解析度自 256 至 40 一部份都切掉了。因此，雖然資料有 256 個不同值，但顯示却僅能容納 40 個不同值。將資料值除以 6.5 就是要將其“縮短”在這個現有的顯示空間內。您同時亦會發現，零的資料值亦使螢幕亮出一個方格。遺憾的是，不論 X 在螢幕的那裡，執行 HLIN0，0 指令時，BASIC 程式都將產生一個“亮”方格。

這個實驗的重點是，計算機可以許多種方式獲得資訊、顯示資訊，或使用資訊。輸入口與輸出口僅是一些將資訊輸入與輸出計算機的其它方式罷了。

實驗11

簡易數位至類比轉換器

目的

這個實驗說明如何將一簡單八位元數位至類比轉換器（

DAC 或D/A) 界面至Apple。

討論

實驗將一簡單之D/A轉換器—— Signetics NE5018 八位元轉換器—— 界面至Apple。雖然我們一直未討論過類比轉換器，但在實驗步驟中我們會做些必要的說明，若有必要，作者希望您進一步參考諸如"微電腦類比轉換器軟體與硬體界面"等有關之書籍。其它主題，諸如取樣與持住放大器，類比多工器與儀器放大器等，在本實驗中均有討論。

IC接腳圖(圖 6 －11)

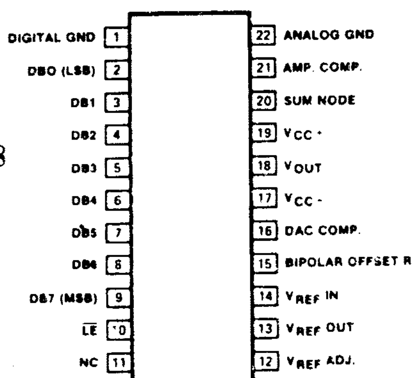


圖 6 －11 Signetcs NE 5018 八位元D/A轉換器晶片接腳圖

第1步

這個實驗另外再需兩個電源，分別爲+12與-12 伏特。這兩個電源將用以供應D/A轉換器。實驗前，一定要先將這兩個電源準備好，並調至適當的電壓。

接好如圖6-12所示的電路。設備選取脈衝取自前面實驗之SN7402 NOR閘電路。這個信號可自圖6-4電路之C點取得，但必須先經過轉換，才能爲NE5018晶片所用。如圖6-13所示的，此一轉換可以一SN7404反相器晶片達成。亦接

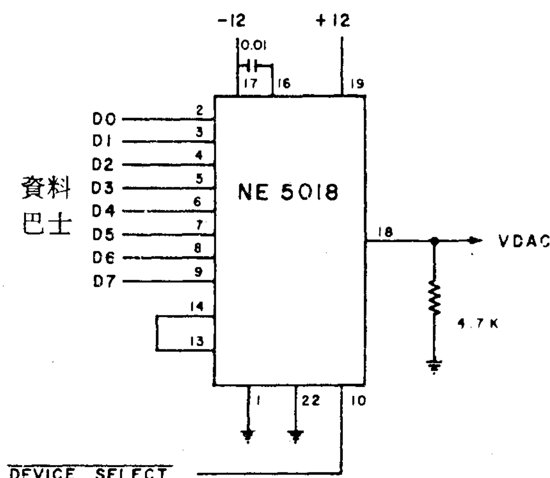


圖6-12 使用NE5018 D/A轉換器晶片之簡單D/A轉換器界面

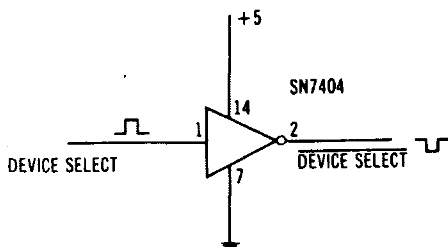


圖 6－13 簡易設備選取脈衝反相電路

好此一反相器電路，將 SN7404 反相器之輸入接至 SN7402 之 13 脚，並將 SN7404 之輸出接至NE5018 轉換器之設備選取（ DEVICE SELECT ）輸入。

此時，小心檢查＋12 與－12 伏特之電源連接，看是否接對。若您用的是分離電源，切記，所有電源之間與麵包板之間必須有共同的低阻抗接地連接。

第 2 步

NE5018 D/A 轉換器會將 0 至 255 之間的數值轉換成 0 至＋10 伏特之間的電壓。由於 0 至 10 伏特之範圍已分成 256 個值，或 255 個階段，因此，電壓增值爲：

10 伏特 / 255 階段＝ 39 m伏特 / 階段

您或許可寫一個增值爲八位元並將之輸出至D/A轉換器的程式。目前，暫且不管D/A轉換器之內部動作情形，就將

之當成一輸出口就是了。您所設計的程式會產生一緩慢上升的正電壓斜波。將您的程式設計在下面的空白裡：

我們使用以下的程式：

```
10  FOR V = 0 TO 255
20  POKE 43918,V
30  NEXT V
40  GOTO 10
```

您可以一簡單的電壓表或三用電表測量輸出電壓值。將這個電表置於接地端與NE5018　VDAC輸出端（VDAC爲正）之間。試試您的程式。電壓會慢慢上升嗎？當電壓抵達+10伏特左右時，情形如何呢？

電壓慢慢上升至+10伏特，在抵達+10　伏特時，又迅速變爲零伏特，或接地，然後再緩慢上升。

您可在程式之內加上一簡短的延時廻路，使電壓上升的速度更慢。這個延時廻路可爲：

```
25  FOR T = 0 TO 100: NEXT T
```

第 3 步

設計一能產生負向斜波的程式，以及一能產生三角斜波（慢慢上升然後又慢慢下降至零）的程式。

這兩個程式可為：

負斜波
```
10  FOR V = 255 TO 0 STEP -1
20  POKE 49318, V
30  NEXT V
40  GOTO 10
```

三角波輸出
```
10  FOR V = 0 TO 255
20  POKE 49318, V
30  NEXT V
40  FOR V = 254 TO 1 STEP -1
50  POKE  49318, V
60  NEXT V
70  GOTO 10
```

您或許想試試這兩個程式其中一個，或您自己所設計的程式。您知道為何三角波輸出之迴路由 254 變化至 1 而非由 255 變化至 0 嗎？

因為，若迴路控制變數值由 255 變化至 0 ，則這兩個值將輸出兩次，雖然您從電壓表上不一定看得出來。這兩個程式亦很值得加上延時迴路。

第4步

您知道0至 10 伏特的電壓正好相當於0至255個階段，您能設計一個使您能自鍵盤打入一個電壓，並使這個電壓顯示於電壓表上的程式嗎？將您所設計的程式寫在下面空白內。

我們設計出下面這個程式，您可上機試試看：

```
10  INPUT "VOLTAGE ";V
20  R = V * 25.5
30  POKE 49318, R
40  GOTO 10
```

第5步

亦試試您自己所設計的程式看看。其亦自D/A轉換器產生一個幾乎同於您所打入之電壓嗎？將電表之誤差考慮在內，我們所提供的程式似乎還不錯。由於這個程式無任何“錯誤偵測”步驟，因此，您可試試由轉換器產生 +15伏特的信號看看。您想結果會如何呢？轉換器會燒掉嗎？

轉換器不致燒掉，因爲，其僅能接受一八位元的值，這個值即相當於+10伏特的輸出。當我們輸入15伏特之" 15"時，由於我們送了一個382的值給八位元設備，因此，會發生"輸入值不合之錯誤"（ILLEGAL QUANTITY ERROR）。

第6步

此時，你應能設計一個使您能打入上限電壓與一下限電壓，並令Apple產生一介於這兩者間之三角波的程式了。使盡您最好的程式設計技巧。

我們採用下列的程式：

```
10  INPUT "UPPER VOLTAGE"; H
20  IF H <= 10 AND H >= 0 THEN 40
30  PRINT "VOLTAGE OUT OF BOUNDS": GOTO 10
40  INPUT "LOWER VOLTAGE"; L
50  IF L <= 10 AND L >= 0
60  PRINT "VOLTAGE OUT OF BOUNDS": GOTO 40
70  IF H > L THEN 90
80  PRINT "UPPER V MUST BE HIGHER THAN LOWER V": GOTO 10
90  H = H * 25.5: L = L * 25.5
100 FOR V = L TO H
110 POKE 49318, V
120 NEXT V
130 FOR V = H-1 TO L+1 STEP -1
140 POKE 49318, V
150 NEXT V
160 GOTO 100
```

執行並測試您所設計的程式。電壓指針應能在上限電壓與下限電壓之間“擺動”。若想看得更清楚一點，您可在程式內加入一延時步驟。

這個實驗清楚地告訴您，如何將一簡單之D/A轉換器界面至您的計算機。NE5018使用了內部鎖住器，而且轉換器晶片上亦置了許多類比電路。D/A轉換器可用於計算機必需控制諸如伺服馬達，放大器等以電壓為主之設備的場合。

以後的實驗將用不到NE5018 D/A轉換器了。因此，您可將這個電路自麵包板拆掉。但SN7402 NOR閘晶片必須留著，SN7404反相器亦可拆開。電源可以關掉。小心拆掉+12伏特與−12 伏特之電源，記得切勿碰到任何其它電路。

實驗12

輸出口，BCD，與二進碼

目的

此實驗的目的乃在探討以SN74LS373晶片作輸出口的用法。

討論

諸如SN74LS373八倍鎖住器等較新的積體電路，可簡化輸出口的構成。這個實驗即以這些晶片其中之一構成一八位元之輸出口，並探討BCD（Binary-Coded Decimal，二進寫碼十進）數之使用。

IC接脚圖(圖6－14)

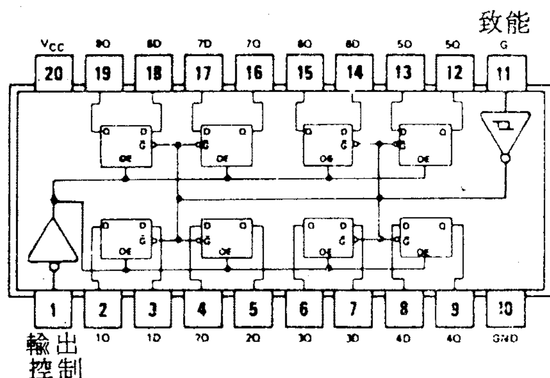


圖6－14 SN74LS373八倍鎖住器晶片之接脚圖

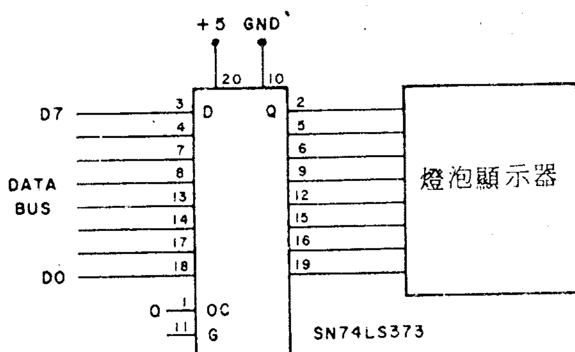


圖6－15 以SN74LS373八倍鎖住器晶片構成輸出口

表6－3　SN74LS373控制信號之眞值表

輸出控制	致能 (G)	資料	輸出
L	H	H	H
L	H	L	L
L	L	X	Qo
H	X	X	Z

第1步

接好圖6－15所示的電路。您可以圖6－4所示之NOR閘電路的13脚，" C "輸出，作爲SN74LS373晶片之" G "輸入。若這個NOR閘電路現在不在麵包板上，請參考實驗2，將之接好。

第2步

注意SN74LS373有兩個控制輸入：G與OC。G輸入控制鎖住，OC輸入控制三態之鎖住輸出。是以，這個鎖住器不僅可用以自巴士獲取資訊，亦可用以將資訊送給另一個巴士。表6－3所示即爲各信號間之關係。

當輸出控制(OC　)信號爲邏輯1時，表示鎖住器輸出已被禁能，或置於高阻抗狀態(HI-Z)。當致能或閘控輸入(G)爲邏輯1時，出現於D輸入之資訊即通過鎖住器電路，到達Q輸出。您可發現，一切動作原理與先前所討論過的SN7475鎖住器晶片完全相同。

做這個實驗時，ＯＣ輸入應接地（邏輯０），使輸出永遠致能（能動作之意）。

第3步

輸出口一接好後，即設計一讀取鍵盤所輸入之數值，並將之以二進形式顯示於輸出口的程式，並加以測試看看。您亦可以二進計數值遞增程式測試這個輸出口。像這些程式您應該都自己會寫了。

第4步

將下列程式打入計算機，並令計算機開始執行。

```
10  FOR C = 0 TO 255
20  POKE 49318, C
30  FOR T = 0 TO 500: NEXT T
40  NEXT C
50  GOTO 10
```

您看到LED如何？

LED 應顯示出一緩慢累增之二進計數。若延遲時間欲增長，您可改變30號之述句。

當 LED 正在顯示一逐次累增之二進計數值時，小心將

SN74LS373晶片之OC輸入(第1脚)與接地端間的接線拿掉，看 LED 顯示有何影響？再接回這條線時，您又看到什麼？

取掉接線時，所有LED均熄掉。當OC輸入又接地時，二進計數繼續。輸出控制(OC)信號並未影響計數。即令輸出被禁能且置於高阻抗狀態，二進計數仍然繼續，且鎖住器亦被計算機"更新"成新的資訊。在目前我們所使用的系統內，輸出之高阻抗狀態使 LED 熄滅。您所看見的或許不同，但您應看到，當 OC 輸入接脚不接地時，鎖住器輸出變化得很厲害。

由於八個鎖住器上均具有三態輸出，因此，SN74LS373晶片特稱爲三態八倍鎖住器晶片。由於含有八個鎖住器，且輸出可置於高阻抗狀態，因此，這個晶片在計算機界面電路上特別有用。SN74LS373可用於連接至數個不同計算機巴士的複雜界面上。事實上，其可作爲可連接兩部或更多部計算機之通信電路的一部份。

第5步

現在您的麵包板上有另一個輸入口，我們再以之探討 Apple 上可做的一些運算。在前面的例題中，我們一直以計

算機控制一累增之二進計數。事實上，二進制並非數位電子儀器所使用的唯一電碼。另一種分別以四位元之二進等效代表每一位十進數字的數目表示法，稱爲**二進寫碼十進**（Binary-Coded Decimal ，簡稱BCD）數，在電子儀器界亦很常用。當然，就每一位元同樣僅可能有兩種值而言，這種電碼還是"二進制"。舉個例子而言，十進數9530 在BCD即表示成1001 0101 0011 0000 。注意到每四位元之間均有隔開。這是表示每四位元本身"自成一個單元"，代表一位十進數字，其與鄰接四位元間之關係並非如一八位元二進數之高次四位元與低次四位元間的關係。許多電子設備均使用BCD碼。七段顯示器（Seven-segment display）與其它許多十進的設備亦都使用BCD碼控制。

試設計一個將一數目"分成"成其 BCD 等效的程式。以輸出口顯示出所有 BCD 數字，每次兩位數字。第一次顯示十位BCD數字與個位BCD數字。接著顯示千位與百位BCD 數字，如此類推。您可利用RETURN 或其它按鍵，令計算機在顯示每兩位數字之間有間歇。

我們設計給您的程式爲：

```
  10  INPUT "VALUE "; A
  20  IF A < 10000 THEN 30 ELSE 10
  30  GOSUB 1000
  40  POKE 49318, A+C
  50  GET A$:A = B
  60  GOSUB 1000
  70  POKE 49318, A+C
  80  GOTO 10

1000  B = 0: C = 0
1010  IF A > 99 THEN 1100
1020  IF A < 10 THEN RETURN
1030  C = C+16: A = A-10
1040  GOTO 1020
1100  A = A-100: B = B+1
1110  GOTO 1010
```

副程式中的變數爲 A ， B 與 C 。 A 代表欲轉換成 BCD 之十進值（即起始值），B 代表百位，C 代表十位。副程式結束時，A 即表個位。願意的話，個位您亦可以另一個變數代表。

有時，您很難記得說您正在教Apple產生 BCD 值，因爲，您所看到呈現於輸出口的一直是二進碼。因此，當您教Apple輸出1001　1001代表99時，Apple 眞正所想的是，其正在輸出十進數153，因爲，LED上所出現的1001　1001　十進等效剛好爲153。所以，您會發現，計算機經常會不知道您教它所顯示的數目是代表什麼意義。

若您想繼續做下一個實驗，請將SN74LS373輸出口留在麵包板上。不過，若您已有了另外的輸出口，SN74LS373電路即可拿掉。電源可暫且先關掉。

實驗13

輸出口紅綠燈控制器

目的

此實驗的目的在說明如何以Apple計算機作實際應用之控制器（ controller ）。

討論

雖然紅綠燈的控制似乎不像計算機之實際應用，但其却顯示了計算機作決策與控制外部事件的能力。

第1步

這個實驗使用一八位元的輸出口。若您的計算機上已接了一個，只要能控制某些 LED ，您即可採用。若您已做完其中之一輸出口實驗，您即可使用在該實驗中所用過的輸出口電路。倘若您欲重新構成輸出口，請您參考實驗 8 之電路。

紅綠燈可以燈泡顯示或個別 LED 模擬。由於南北道可使用同一組燈，東西道亦然，因此，總共僅需六個 LED 就夠了；每一方向三個，紅、黃、綠燈各一個。假設我們使用彩色的 LED ，並採用下述的規則：

位元	LED	
D0	紅	忠孝路
D1	黃	忠孝路
D2	綠	忠孝路
D3	紅	敦化路
D4	黃	敦化路
D5	綠	敦化路

第2步

現在，您必須決定使每一個 LED 個別亮或滅所需的邏輯 0 與邏輯 1 組合型態。我們的電路以鎖住器晶片直接推動 LED，且邏輯 0 使 LED 亮，邏輯 1 使 LED 滅。根據這些

以及上一步所定的規則，您能寫出教每一個 LED 個別亮或滅的位元型態嗎？

根據以上所述，使各個 LED 發亮的位元組合型態（以及其十進等效值）為：

忠孝路的紅燈	254	11111110
忠孝路的黃燈	253	11111101
忠孝路的綠燈	251	11111011
敦化路的紅燈	247	11110111
敦化路的黃燈	239	11101111
敦化路的綠燈	223	11011111

第3步

為了開始紅綠燈控制作業，我們設計一個使敦化路之黃燈與忠孝路之紅燈閃亮；亮一秒，滅一秒，的程式。試問"亮"的位元型態如何？"滅"的位元型態又如何？

滅之位元組合型態爲每一位元均邏輯1，值255；而亮之位元組合型態爲D4與D0兩位元爲邏輯0，其餘位元均爲邏輯1，值238。程式爲：

```
10  POKE 49318,255
20  FOR T = 0 TO 770: NEXT T
30  POKE 49318,238
40  FOR T = 0 TO 770: NEXT T
50  GOTO 10
```

第4步

求出正常紅綠燈控制所需之燈亮組合型態。總共需幾種型態？爲那些型態？這些型態如何儲存於計算機內？

僅需四種型態：(a)忠孝路紅燈亮，敦化路綠燈亮（值222）；(b)忠孝路紅燈亮，敦化路黃燈亮（值238）；(c)忠孝路綠燈亮，敦化路紅燈亮（值243）；(d)忠孝路黃燈亮，敦化路

紅燈亮（值 245）。這些值可以DATA述句，註標（subscripted）變數，或僅以變數儲存於計算機內。

第5步

這個實驗爾後的部份，我們假設黃燈期間均爲2秒。因此，若忠孝路之週期爲10秒，則其綠燈將亮10秒，然後黃燈緊接再亮2秒，之後，再變爲紅燈。

假設忠孝路之週期爲6秒，敦化路之週期爲 10 秒，試設計一能自動變換這兩個街道交叉口上之紅綠燈的程式。

我們提供給您的程式爲：

```
10   M = 10: E = 6: P = 49318
20   DATA 222, 238, 243, 245
30   READ L
40   POKE P,L
50   FOR R = 1 TO M
60   FOR T = 0 TO 770: NEXT T
70   NEXT R
80   READ L
90   POKE P,L
100  GOSUB 1000
110  READ L
120  POKE P,L
130  FOR R = 1 TO E
140  FOR T = 0 TO 770: NEXT T
150  NEXT R
160  READ L
170  POKE P,L
180  GOSUB 1000
190  RESTORE
200  GOTO 30

1000 FOR R = 1 TO 2
1010 FOR T = 0 TO 770: NEXT T
1020 NEXT R
1030 RETURN
```

第6步

前一步所列的程式雖然能正確運作，但其中有許多步驟却是重複出現的。您能將這個程式改寫得更簡單一點嗎？您怎麼做的呢？

原有程式之四個基本片段，唯一改變的僅有延遲時間與燈亮型態。因此，若採用陣列儲存所有的延遲時間與燈亮型態，原來程式之四個片段即可寫成一個廻路。經過這樣修改後的程式即如：

```
 10  A(1) = 222: A(2) = 238: A(3) = 243: A(4) = 245
 20  M(1) = 0: M(2) = 2: M(3) = 0: M(4) = 2
 30  INPUT "MAIN DELAY "; M(1)
 40  INPUT "ELM DELAY "; M(3)
 50  FOR Q = 1 TO 4
 60  POKE 49318, A(Q)
 70  FOR R = 1 TO M(Q)
 80  FOR T = 0 TO 770: NEXT T
 90  NEXT R
100  NEXT Q
110  GOTO 50
```

這個新的程式以陣列 A 儲存燈亮型態，陣列 M 儲存時間間隔。

第 7 步

截至目前爲止，計算機所扮演的角色僅是一循序器，依次產生適當的燈亮型態與時間延遲。這個步驟將在紅綠燈控

制程式內加入一些控制步驟。

由於敦化路之交通經常較擁擠，故紅綠燈之狀態正常應敦化路綠燈亮且忠孝路紅燈亮。新的程式應能測知忠孝路是否有車在等，若有一輛車在等，即使其綠燈亮。不過，於測知忠孝路有車前，敦化路至少應有 30 秒鐘之"綠燈時間"。這個意思是說，並非忠孝路上每一輛等候的車輛均能自動引發一忠孝路綠燈亮系列。為了使情況變得更有趣一點，我們在敦化路上亦設一個察覺器。若敦化路上紅燈時有五輛或五輛以上的車在等，這條路即應變綠燈亮，而忠孝路必須改紅燈亮。

為了寫出程式，您或許可先畫出一簡單的電路圖。兩條道路上察覺器可以一輸入口加以模擬。不過，為了讓您多學一點，這裡我們將採用鍵盤。

Apple之鍵盤使用兩個記憶位址作控制。位址 49152含鍵盤資料，而位址 49168作為旗號清除脈衝輸出。

將下列程式打入計算機，並加以執行：

```
2000  PRINT PEEK(49152): GOTO 2000
```

在鍵盤上隨意按一些鍵，看顯示會有何改變。您看到什麼呢?

每次按一個新鍵，即有一個新的十進值顯示出，而且這個新

值一直顯示至您再按下一個新鍵時爲止。由此可見，輸入口 49152上的資訊即代表最後一個按鍵的電碼。

第 8 步

我們希望計算機唯有在當鍵盤上有按鍵時，才自鍵盤輸入口輸入一個數值。爲了達到這個目的，我們必須設置一個鍵盤旗號位元，這個位元就是 49152 輸入口之D7 位元。倘若這個位元爲邏輯 0 ，由這個輸入口所輸入的所有數值即小於 128 。若此位元爲邏輯 1 ，則該輸入口所輸入的數值即等於或大的 128 ，但最高 255 。因此，只要測試該輸入口所輸入的數值，您即可知道鍵盤上是否有鍵按下。當然，在“測知”有按鍵之後，您必須以一讀取 49168的讀取作業，將旗號位元清除爲零。

打入且執行下面的程式：

```
2000  IF PEEK(49152) >= 128 THEN PRINT PEEK(49152)
2010  Z = PEEK(49168)
2020  GOTO 2000
```

現在，在鍵盤上隨意按幾個鍵，每次一個。請問顯示如何？在您按下鍵時，每一個按鍵的十進碼都顯示於顯示器上嗎？

或許，您會發現有些按鍵好像“漏掉”了，未顯示出來。這

是因爲計算機每次執行過廻路，即將鍵盤旗號清除，因此，很可能 Apple 清除了鍵盤旗號之後，程式才測知有無按鍵。理想上，旗號應在程式已測知有按鍵之後，才能將旗號清除。

第 9 步

設計一簡短的鍵盤控制程式，使其能測知每一個按鍵（絕不漏掉），而且每一按鍵僅測知一次，並且印出按鍵之等效十進值。

我們使用下面不斷地檢查鍵盤，並於唯有查到旗號值爲 1 時才印出一個文字，然後唯有在這個時候才清除旗號的程式：

```
2000  IF PEEK(49152) < 128 GOTO 2000
2010  PRINT PEEK(49152)
2020  Z = PEEK(49168)
2030  GOTO 2000
```

注意，變數 Z 乃一虛擬變數，其唯一作用就是使 PEEK (49168) 命令能清除鍵盤旗號。

若您希望進一步使用按鍵之十進值，則您可將輸入值減去 128 ，去掉旗號位元。

第10步

寫出您的紅綠燈控制程式，並作測試，以“ E ”鍵代表忠孝路之察覺器，“ M ”鍵代表敦化路之察覺器。當然，您必須知道這兩個按鍵的對應鍵碼。

以下即為我們列給您的程式。為了測試，這個程式使用一近於 10 秒之週期，以及 2 秒之黃燈週期。

```
10  A = 0: P = 49318
20  POKE P, 222
30  FOR R = 0 TO 10
40  FOR T = 0 TO 770: NEXT T
50  NEXT R
55  Z = PEEK(49168)
60  IF PEEK(49152) = 197 GOTO 80
70  GOTO 60
80  Z = PEEK(49168): POKE P, 238
90  FOR R = 1 TO 2
100 FOR T = 0 TO 770: NEXT T
110 NEXT R
120 POKE P, 243
130 FOR R = 0 TO 1000
150 IF PEEK(49152) = 205 THEN 190
170 NEXT R
180 GOTO 210
190 Z = PEEK(49168): A = A+1
200 IF A < 5 THEN 170
210 POKE P, 245
220 FOR R = 1 TO 2
230 FOR T = 0 TO 770: NEXT T
240 NEXT R
250 GOTO 10
```

您應發現，一開始，50 號之述句先將旗號值清除為零，然後 60 號之述句再測知旗號值。此舉清除了最初 10 秒內的任

何按鍵輸入。若您希望忠孝路之察覺器能"記住"在這段期間內行經這條路的車子，則您可除去這個步驟。

150 號之旗號測知步驟已插入整個計時廻路中。這表示旗號一直不斷被檢查，而且這些旗號檢查步驟必須計入整個延遲時間內。以各種延遲常數測試一下 130 號之述句，您即可知曉。

這個程式尚可做其它許多事情。譬如，許多交叉路口均有人行控制信號，左轉信號，閃爍燈，與其它特殊特色等。您可將程式變得如您所想像地複雜。在這種情況下，時序並非極度重要。在測試旗號時，萬一排隊等候的車多等了一、兩秒，亦不甚要緊。不過，若 10 秒或 20 秒，則駕駛先生小姐們可能就會惱火了。因此，在設計程式時，切記這一點。有時，時間要求必須很嚴格，而且時間週期亦很短暫，這種情況下，最好就必須使用組合語言設計程式。

將六個 LED 拆掉，輸出口留著，電源亦可關掉。

實驗14

邏輯元件測試器

目的

本實驗的目的乃在說明如何以計算機測試電子元件。譬如，測試簡單的邏輯閘。

討論

絕大多數含邏輯閘的邏輯晶片，均可以在其輸入加上已知邏輯準位，然後將其輸出與元件之眞值表相比的方式加以測試。這個實驗則以計算機來做。實驗需要一個輸入口與一個輸出口。如SN7400，SN7402，SN7408，等等之各種元件均可受測試。這種測試是一種功能測試，而非諸如交換時間、傳遞延遲、與其它參數之動態特性測試。

第1步

這個實驗必須用到一個輸入口與一個輸出口。請按前面幾個實驗所提供的資料，準備好這兩個口，輸入口可使用SN74LS373晶片。當這些口均接好與測試好時，即進行下一步驟。

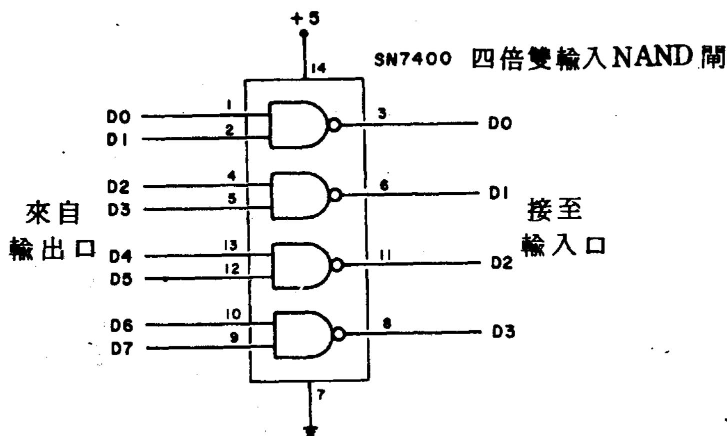


圖6－16 SN7400 NAND閘測試電路圖

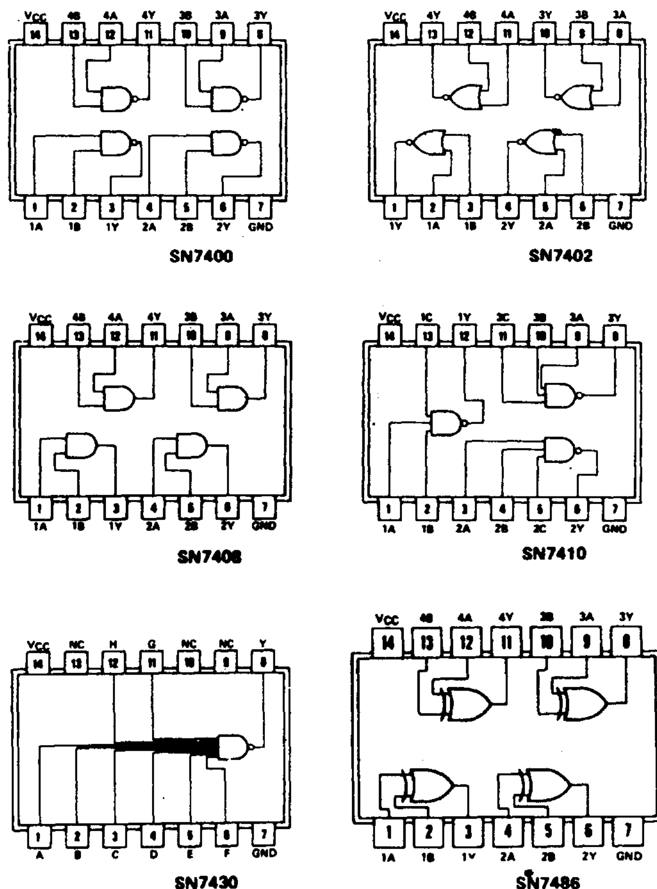


圖 6－17　一些標準邏輯閘之接腳圖

第 2 步

SN7400　NAND閘之測試電路如圖 6 － 16所示。其它晶片之接腳圖則如圖 6－17 所示。

接好如圖 6－16 所示的測試電路。記得界面晶片與被測試電路兩者都必須接＋5伏特與接地，輸入口未用的輸入亦應接地。

由 NAND 閘開始，您應能自行找出圖 6－17所示之各種邏輯閘的眞值表。就兩輸入之邏輯閘而言，所有輸入總共才僅有四種組合。同一塊 IC 內有同樣邏輯閘者，總共有多少組合呢？

很可能您會說有 16 種組合，因爲，四個閘每個閘四種；或 256 種組合，因爲，可能輸入數有八個。實際上，由於每一個閘均完全相同而且同時測試，所以，眞正有意義的組合僅有四種。求知某一個閘在某種輸入組合時反應不良並無多大用途。因爲，若一個閘不良，則整塊 IC 就等於不良了。

第 3 步

測試 NAND閘時，輸出口八位元之四種組合是那四種呢？分別寫下這些數的十進與二進值。

這些值乃：

```
00  00  00  00  =   0
01  01  01  01  =  85
10  10  10  10  = 170
11  11  11  11  = 255
```

由於四個輸出接至D3～D0等輸入位元，因此，我們想這些若非全為1，即全為0，換言之，數目一定等於0或 15。為了“除去”不用的位元，我們將D7～D4 接地。這幾個輸入的輸入值如何呢？其會影響到結果嗎？您能提出另一種“免除”這些位元的方式嗎？

接地的幾個位元均輸入邏輯0，且其不應影響資料。若不先將這些位元接地，則事後以一邏輯 AND 運算將之遮掉亦可。實施邏輯 AND 運算時即可用到前面介紹過的組合語言副程式。

第 4 步

設計一能測試您已接好之NAND閘的簡短程式。這個程式可能極類似於實驗 13 之第 6 步所示的紅綠燈控制程式，其並不須很複雜。

下面是我們所設計的程式：

```
10  T(1) = 0: T(2) = 85: T(3) = 170: T(4) = 255
20  R(1) = 15: R(2) = 15: R(3) =15: R(4) =0
30  FOR S =1 TO 4
40  POKE 49318, T(S)
50  IF PEEK(49319) <> R(S) THEN 100
60  NEXT S
70  PRINT "TEST OK": END
100 PRINT "FAILURE": END
```

第 5 步

SN7400，SN7408 與 SN7486的接脚圖均相同，亦即，輸入與輸出均在晶片之相同位置上，試問能因而設計一個能測試這每一個晶片的一般化程式嗎？怎麼設計呢？

是的，可設計一個一般化的程式，用者只要打入元件名稱，計算機即可建立實驗中用到的適當眞值表資訊。這些眞值表即如表 6 － 4 所示。

表 6 － 4　NAND, AND, 與 EXOR 閘之眞值表

SN7400			SN7408			SN7486		
A	B	OUT	A	B	OUT	A	B	OUT
0	0	1	0	0	0	0	0	0
0	1	1	0	1	0	0	1	1
1	0	1	1	0	0	1	0	1
1	1	0	1	1	1	1	1	0

您應發現，測試類型完全相同，唯有結果改變。

我們採用下面的測試程式：

```
 10  INPUT "LAST TWO DIGITS ";A$
 20  IF A$ = "00" THEN 200
 30  IF A$ = "08" THEN 300
 40  IF A$ = "86" THEN 400
 50  PRINT "TEST NOT AVAILABLE": GOTO 10
 60  T(1) = 0: T(2) = 85: T(3) = 170: T(4) = 255
 70  FOR S = 1 TO 4
 80  POKE 49318, T(S)
 90  IF PEEK(49319) <> R(S) THEN 120
100  NEXT S
110  PRINT "TEST OF SN74";A$;" OK":END
120  PRINT  "FAILURE": END
200  R(1) = 15: R(2) = 15: R(3) = 15: R(4) = 0
210  GOTO 60
300  R(1) = 0: R(2) = 0: R(3) = 0: R(4) = 15
310  GOTO 60
400  R(1) = 0: R(2) = 15: R(3) = 15: R(4) = 0
410  GOTO 60
```

程式最後所要求的兩位數字即爲元件代號之最後兩位數字；譬如，SN7400 即爲 00 ，SN7408 即爲 08等等。若手邊同時有數個 SN7400，SN7408 或 SN7486 晶片，則您可一一都加以測試一遍。您亦可試著取掉某一條輸入線或輸出線，模擬一下錯誤狀況，測驗一下您的界面與程式看看。

第 6 步

計算機亦應能測試正反器與計數器等邏輯元件。若您熟悉 SN7493 四位元二進計數器，則您可試一下下面的步驟。萬一您不熟，您亦不妨讀一讀。

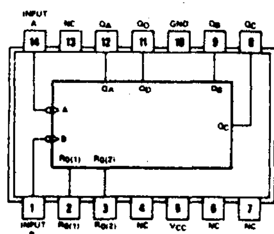


圖 6－18 SN7493四位元計數器之接脚圖

SN7493 計數器的接脚及電路圖如圖 6－18 所示。為了測試這個元件，計數器輸出必須加至計算機，且計算機必須能重置（ reset ）與時序推動計數器晶片。我們將僅測試重置計數器的能力以及計數的功能，而不做全面徹底的測試。

第 7 步

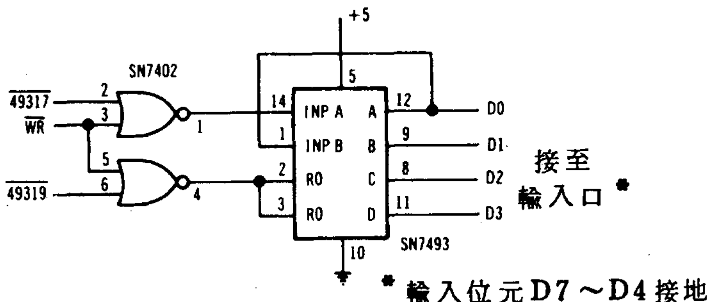


圖 6－19 檢查SN7493計數器晶片所用的測試電路圖

將SN7493計數器如圖6－19所示的接好。這回必須用到這個實驗前幾個步驟中所使用的輸入口與輸出口。此外，如圖6－19所示的，還需要兩個NOR閘。這兩個NOR閘只要一個SN7402晶片就夠了。記得勿以SN74L93計數器代SN7493。輸入口中未用的輸入亦記得接地。

第8步

設計一能重置計數器，並能測試計算機時序推動計數器且將計數器之計數值加1之能力的簡短程式。

我們提供您下面的程式：

```
10   POKE 49318,0
20   IF PEEK(49319) > 0 THEN 1000
30   PRINT "RESET TEST OK"
40   FOR C = 1 TO 15
50   POKE 49317, 0
60   IF PEEK(49319) <> C THEN 1010
70   NEXT C
80   PRINT "COUNT TEST OK": END
1000 PRINT "RESET FAILURE":END
1010 PRINT "COUNT FAILURE AT "; C: END
```

這個程式先測試重置，然後再開始測試晶片在 INP A 接脚每收到一個脈衝時，將計數值加 1 的能力。

第 9 步

這個程式並未測試到全部 16 種計數器狀態。最後一次由 1111 至 0000的計數並未測試到。請問您能修改程式，使其涵蓋這次計數嗎？
在程式加上這個最後的測試並不難。有數種方式可以達成目的。以下即是其中一例：

```
90  POKE 49317,0
100 IF PEEK(49319) <> 0 THEN 1010
110 PRINT "COUNT TEST OK": END
```

加上這幾個步驟後，計數器即可產生最後一次的計數。並且由 1111 至 0000的"回數"亦能測試到。

輸出口不再用，因此，可以自麵包板拆掉。但輸入口還要用。程式不再用了，因此，電源可以關掉。

實驗15

簡易旗號電路

目的

本實驗的目的乃在說明如何構成與使用簡單的旗號（flag　）電路。

討論

旗號是計算機與輸入／輸出設備爲了同步彼此作業所使用的信號。每一個旗號通常代表兩種可能的狀態，譬如：預備好／忙碌，滿／空，熱／冷，或其它與計算機之界面有關的狀況組合。本章之實驗6曾做過以輸入口傳送非數值資訊給計算機。這個實驗將進一步擴充這個觀念。實驗用到一個八位元的輸入口。

IC接脚圖（圖6－20）

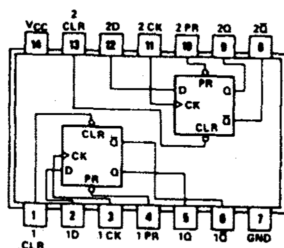


圖 6－20　SN7474雙倍 D型正反器晶片之接脚圖

第 1 步

這個實驗需要一個輸入口。此刻，您應毫無問題能自己構成這個輸入口。本章前面許多實驗都用過這種輸入口，我們建議您取其中之一而用。輸入口一經接好與測試好後，即繼續進行下一個步驟。

第 2 步

前面有一個實驗曾以簡單的開關作爲察覺器或旗號輸入。這個實驗我們將以正反器取代機械式開關或跳線。接好如圖 6－21 所示的電路圖。

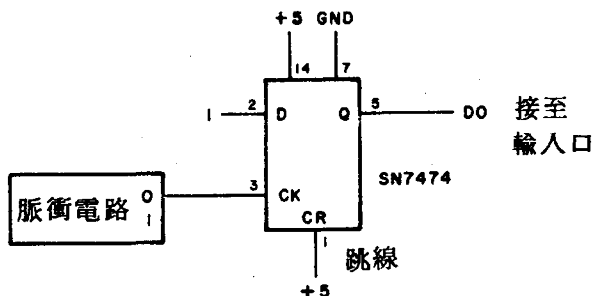


圖 6 — 21　以正反器構成之簡易旗號電路

＋ 5 伏特與清除輸入（第 1 脚）間應以跳線作連接，以便欲清除旗號時，您能直接取掉＋ 5 伏特至地之間的接線，然後再接回去即可。圖中所示的脈衝電路可爲兩個交錯交連的NAND閘，或一能產生完全無雜訊之邏輯轉換的相等電路。這種功能於附錄中有介紹。

第 3 步

您如何設計程式敎計算機監聽著輸入口之 D0 位元的邏輯準位？假設有兩種可能狀況：(a)其它位元均接地（邏輯 0），或(b)其它位元可用作其它旗號輸入。

若其它位元接地，則當旗號清除時，輸入口所輸入的值將爲

零；而旗號被置定為 1 時，輸入口所輸入的值不為零。若其它位元用作旗號輸入，則“不要”的位元必須遮掉（ masked ）。遮蓋作業可透過叫用組合語言副程式，用邏輯AND運算達成。

第 4 步

這個實驗，您必須打入對兩個資料位元組作 AND 運算的組合語言程式。請遵照下列步驟將這個程式打入計算機：

1. 按RESET鍵，打CALL－151 ，然後按RETURN鍵。處於監督程式（ monitor ）型態時，Apple 應顯示出星號（ * ）。

2. 自鍵盤打入0300 ： 00 00 00 48 AD 00 03 2D 01 03 8D 02 03 68 60 。每一組兩位數字打完後，請均留一個空格。以 00 作程式的最前面三個值。

3. 按RETURN鍵，打入02FF，並連按RETURN鍵三次。這時，檢查一下螢幕上所顯示的訊息是否與您欲打入者相同。

您可以下面的程式測試這個組合語言副程式。由於AND運算使用二進數，因此，您亦須將您的測試數目轉換成二進制，以便能核對結果。

```
1000  POKE 10,76: POKE 11,03: POKE 12,03
1010  INPUT " MASK BYTE ";M: POKE 768,M
1020  INPUT " DATA BYTE ";D: POKE 769,D
1030  Q = USR(0): PRINT "ANSWER "; PEEK(770)
1040  GOTO 1010
```

若這個程式無法產生正確結果，即令Apple計算機回至監督程式型態，再檢查一遍您所打入的資料位元組。

您該知道，1000 號 POKE 指令之功用在設定指示器位址位元組，使 USR 指令能找到您所打入之組合語言副程式的位置。有關這個組合語言程式的用法與其它細節，請參考第 4 章與實驗 7 。

第 5 步

對兩個資料位元組作 AND 運算，並產生一八位元結果的組合語言副程式已打入計算機，現在，您可以之開始測試旗號位元了。請問您拿什麼作面罩位元組？

由於旗號由 D0 位元輸入至計算機，唯有最低次位元（LSB）會被設定爲 1 ，因此，面罩位元組應爲 00000001_2 或 1_{10} 。由前一步驟之測試程式您可輕易看出，面罩位元組被置於位址 768 之記憶位置內。

第 6 步

設計一個能測試正反器旗號電路的程式。若旗號清除，Apple 應印出" 0 "，否則，若旗號被設定為 1 ，程式應教 Apple 印出" 1 "。欲清除旗號時，您可以手除去連接正反器之第 1 脚與＋ 5 伏特間的跳線，使第 1 脚瞬間接地。

我們提供下面的程式：

```
10  POKE 10,76: POKE 11,3: POKE 12,3
20  POKE 768,1
30  POKE 769,PEEK(49319)
40  Z = USR(0)
50  IF PEEK(770) = 0 THEN 80
60  PRINT "1"
70  GOTO 30
80  PRINT "0"
90  GOTO 30
```

這個程式似乎動作得非常好。您能將這個程式" 倒反 "，使得邏輯 0 被感應為" 動作 "（ ON ）狀態，而邏輯 1 被感應為" 不動作 "（ off ）狀態嗎？

是的，只要將60號與80號之述句對調就可以了。旗號的意義在軟體上很容易"倒反"。

第7步

這個步驟以一個簡短程式計算旗號值被設定爲1的次數。實驗再度用到組合語言副程式。此時，您可考慮另外增加一個脈衝電路來作旗號清除，而不必再以手拿掉SN7474之第1脚與+5伏特間的跳線。

打入並執行下面的程式：

```
10  POKE 10,76: POKE 11,3: POKE 12,3
20  POKE 768,1
30  HOME: C = 0
40  POKE 769, PEEK(49319)
50  Z = USR(0)
60  IF PEEK(770) = 0 THEN 40
70  C = C+1: HTAB 1: VTAB 1: PRINT C
80  GOTO 40
```

記得，在測試程式前一定要先清除正反器。當程式正在執行之際，令脈衝電路動作，將正反器設定爲1。您看到的結果如何？是您所預期的嗎？

我們發現，正反器一被設定爲 1 ，計數即開始。而且只要旗號仍爲 1 ，計數即繼續不斷。一清除正反器時，計數即停止。眞正我們所想要的是：正反器每次被設定爲 1 時，計數值才加 1 。

爲何結果不是我們所預期的呢？因爲，程式的設計是不斷地測試且測知正反器的置定狀態（ 1 ）。而由於我們手的動作太慢，因此，當我們以手欲淸除正反器時，計數廻路不知早已多數過多少次了。

第 8 步

在絕大多數計算機系統上，旗號一經測知後，計算機或含旗號的元件通常都會立即將旗號清除掉。在這種情況下就不致發生前一步驟所言及的問題了。爲了使界面電路能自動淸除旗號，我們加上如圖 6－22 所示的電路。這個電路必須用到一個 SN7402 NOR 閘電路。記得，SN7402 晶片的第

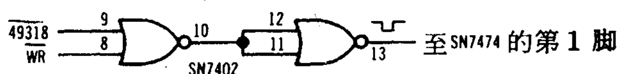


圖 6 －22 簡單的旗號淸除電路圖

14 脚必須接+5伏特電源，第 7 脚必須接地。由於正反器清除信號將由NOR閘電路產生，因此，原來+5伏特與SN 7474之第1脚間的接線必須除去。

加上圖6－22的電路後，只要使用POKE　49318 指令，您即可清除正反器。

將以下65號一列加至原來的程式內：

```
65  POKE 49318,0
```

每次計算機一執行這個命令，旗號即會被清除爲零，由於程式一開始時，正反器之狀態無法得知，因此，您亦可在程式最前面加上一個相同的旗號清除指令。現在，執行程式看看。您會發現，每次旗號被測知（爲1）後，其即立即被清除爲0。然後，計數值再加1，並顯示出。

使用這種旗號以及使用組合語言副程式檢查旗號的好處之一，就是您不必教計算機停著不做事專門來等這個旗號。相反地，您可設計一個程式來檢查這個旗號。每當旗號不出現（亦即，爲邏輯0）時，計算機即做其它的事去。而當旗號出現時，計算機再來服務與旗號有關的設備。

Apple內的BASIC 解釋程式有一個旗號檢查指令稱爲WAIT。這個指令可用以測試旗號。但使用這個指令時，若旗號不出現，計算機會繼續等，不會自動做其它事去。因此，若萬一發生程式"掛在那兒"一直等，而旗號又一直不出現時，您可按下RESET 鍵，令控制重新回至Apple。有關WAIT指令之進一步細節請參考"BASIC 程式設計參考手

册”`。WAIT 指令並未含旗號自動清除作業。

實驗16

簡易類比至數位轉換器

目的

這個實驗以一八位元類比至數位轉換器與計算機界面，並做多種不同的測量。

討論

類比至數位轉換器，或簡稱A/D轉換器，在計算機系統內有許多種用途。A/D轉換器使計算機能測量各種信號源與轉送器（ transducer ）所產生的類比電壓。這個實驗使用一個國際半導體公司之ADC0804 型八位元A/D轉換器。這個轉換器具有三態輸出，因此，其毫無問題可直接界面至微電腦之資料巴士上。不過，由於其三態輸出的存取時間（access time）可能高達 200ns ，因此，若您想將ADC0804 A/D轉換器用於您的界面麵包板上，您會發現，推動巴士內鎖電路，使資料巴士回頭作輸入所需的時間過長，以致計算機將無法讀到轉換器所送出之資料。

因此，爲了做這個實驗，您必須使用“空餘的”Apple

資料巴士。這將在下面幾個步驟中說明。

IC接脚圖(圖6－23)

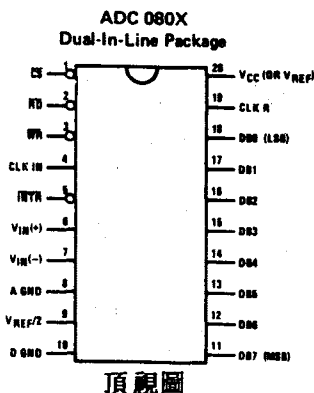


圖6－23 ADC0804 A/D轉換器之接脚圖

第1步

這個實驗將ADC0804 A/D轉換器當成宛如來自Apple一樣，直接接至資料巴士。爲此，請小心取下界面麵包板上IC-10與IC-11上的兩個8216巴士緩衝器晶片。

第2步

將ADC0804 IC如圖6－24的接好。資料巴士線插入IC-10與IC-11兩個插座之對應孔上。若線不太容易插入，我們建議您在IC-10與IC-11兩插座上裝上一個出入孔更大的16脚插座。如此，您即不必再將線硬擠入小小的插

孔內。將線插入插孔時千萬不可用力過甚，否則，您可能會弄彎了插座接點，致使 8216 晶片插入相對插座時，其無法與這些晶片適當接觸。

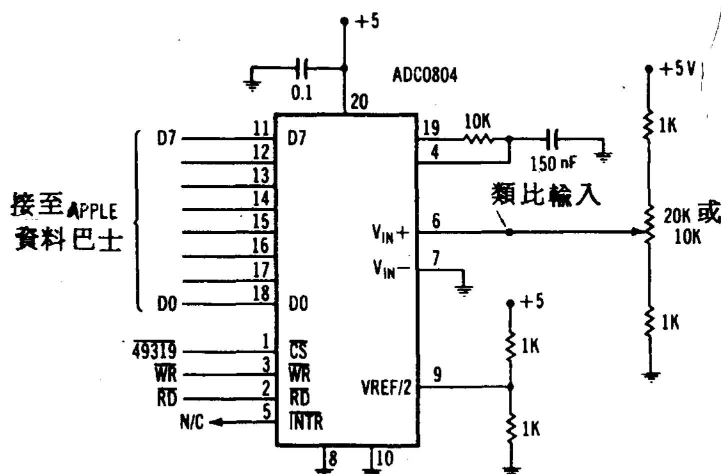


圖 6 —24　ADC0804界面電路圖

第 3 步

將下面程式打入計算機，並教計算機開始執行：

```
10  POKE 49319,0
20  FOR T = 0 TO 100: NEXT T
30  PRINT PEEK(49319)
40  GOTO 10
```

這個程式作何用？顯示螢幕上顯現了些什麼？

程式運用了A/D轉換器；起動轉換程序，產生時間延遲，使轉換能以進行，之後並讀入與顯示資料。慢慢的調整電位計，眼見著資料，直到證實轉換器確實在動作爲止。

當您改變電位計之電壓設定時，您會看到Apple計算機所顯示出的數值亦隨著改變。最小值與最大值各多少呢？與您所想像的相同嗎？

最小值應在 0 或 1 的範圍。最大值則應在 253 至 255 之間。由於八位元的設備僅能產生 0 至 255 之間的數值，因此，這正是我們所預期的。

第 4 步

ADC 0804 晶片有一個可用以測知轉換器之狀態——忙碌或準備好——的旗號輸出。當資料備妥準備輸入計算機時，此一輸出爲邏輯 0，而當轉換器正在作轉換時，該輸出即爲邏輯 1。這個輸出實際上就是旗號電路的輸出，旗號在八位元資料被讀入計算機時自動清除爲零。由於轉換器每秒鐘能做數千次轉換，因此，您想有必要監聽這個旗號信號嗎？

或許不必，因為，在BASIC 語言程式讀取資料之前，轉換器一定早已完成整個轉換程序了。

您能提出這個旗號輸出的一些可能用途嗎？

旗號可用於A/D轉換器之組合語言程式設計。於組合語言程式內，旗號可被當成輸入口之輸入測試，或者，其亦可用於6502 微處理器晶片之插斷（ interrupt ）。由於這些都是屬於高速度的運用，因此，您可能要監聽旗號，求知轉換器何時完成轉換。

第 5 步

除去程式 20 號一列，再執行程式，您發現什麼？

資料值與程式使用延時步驟時所見者完全相同。由此可見，轉換器“勝過”了BASIC控制程式。

第 6 步

螢幕上所顯示的數值並不眞正代表實際被測量的電壓值，而僅爲一八位元二進表示。設計一個能將之轉換成電壓值的程式。您可將這幾個步驟加在原有的程式內。

我們以下列的程式步驟，將 0 至 255 之十進值轉換成 0 至 + 5 伏特之相對電壓。

```
10  POKE 49319,0
20  FOR T = 0 TO 100: NEXT T
30  PRINT (PEEK(49319)*5/255)
40  GOTO 10
```

試一下我們給您或您自己所設計的程式，靈嗎？

當然該靈。您會看到，計算機印出了許多位數，或許太多了一點，因爲，轉換器至多僅能精確至約 0.25 %左右。遺憾的是，四捨五入在Apple 而言並非是一件十分容易的事。您可利用數學式的四捨五入，或採用字串運算，教計算機僅印

出某一固定的小數位就好。願意的話，您可使用下面的述句：

```
30  A$ = STR$(PEEK(49319)*5/255)
40  PRINT LEFT$ (A$,4)
50  GOTO 10
```

記得，這個常式僅將顯示值限制至 4 個小數位，其並未做四捨五入。

第 7 步

試利用 Apple 計算機的高度圖形解像力，設計一個能教計算機畫出電壓對時間值之圖形的程式。程式應畫出連續線，並於每隔一段固定時間（延時程式）作一次測量。若您對高解像之圖形格式與命令不熟，則請您使用下面的程式：

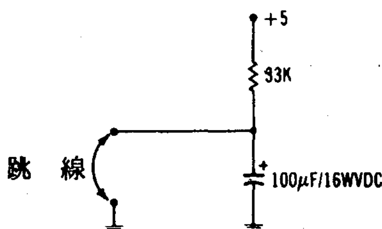
```
100  HRG: HCOLOR = 3: Y1 =0
110  FOR X = 0 TO 249
120  POKE 49319,0
130  Y2 = PEEK(49319)/1.594
140  H?LOT X,Y1 TO X+1, Y2
150  Y1 = Y2
160  NEXT X
170  END
```

試試這個程式，於程式執行過程中變換電位計設定。當您做變換時，圖形應出現於螢幕上。固定電壓在螢幕上將顯示出一水平線。

第 8 步

您能提出一個能說明 A/D 轉換器與圖案程式之用途的簡單實驗嗎？

有好幾個簡單的實驗都可以做。每一個均涉及測量一與正在測之物理量成正比例的電壓。譬如，您可測量當光照亮度改變時，一光敏電阻兩端的電壓，或一正在充電之電容兩端的電壓，或甚至一與溫度成正比之電壓值。



接好如圖 6－25 所示的電路。這個電路以 A/D 轉換器及計算機測量一大電解電容兩端的充電電壓。

使用跳線令電容器放電，並一直放電至程式開始執行時爲止，程式一旦開始執行，即將接地之跳線拿掉。您應看到，在電容器充電之際，電壓緩緩地上升。試問圖形之零電壓點爲何在螢幕頂端，而高電壓在螢幕底端？

因爲，當電壓值由小漸漸變大時，計算機亦由螢幕上緣往下畫，所以圖形看起來是倒向的。若您欲將圖形"端正"過來，您即必須將程式之130號一列改成：

```
130  Y2 = 160-(PEEK(49319)/1.594)
```

以便將0值轉換成159，將159轉換成0。

第9步

A/D轉換器亦可用以測量溫度。您可以一LM335溫度察覺器，產生一與溫度成正比例（比率爲10 mV/K）的電壓。除了0°C = 273K外，凱氏（Kelvin）溫度刻劃與攝氏溫度刻劃全然相同。所以，室溫20°C即等於293K，而LM335將產生293 × 10mV = 2.93伏特之電壓輸出。

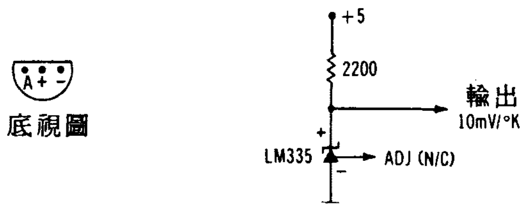


圖 6－26　溫度測量電路圖與 LM 335晶片之接脚圖

接好如圖 6－26 所示之電路。記得，電位計或電容充電電路切勿與溫度察覺器同時接至A/D 轉換器的輸入。

您可直接使用前一步驟用過之圖案顯示程式，亦可在155 列上加入下面的延時步驟：

```
155  FOR T = 0 TO 100: NEXT T
```

由於溫度改變之速度較電容充電電壓改變之速度爲慢，因此，我們以這個常式延長顯示的時間。

執行程式，以手指將察覺器加熱，您是否看到任何變化？您想您會看到什麼呢？

由於 0 至 500 K 的溫度範圍僅轉換成 0 至＋5 伏特的顯示電壓，因此，您可能看不出有多大改變。萬一您看到顯示增加了幾“點”以上，那您一定在察覺器上加了很多的熱。您亦可以一點水氣，或以一罐冷却電子零件時常用的冷噴液，輕易地將察覺器冷却。若這些東西都沒有，可以使用冰塊。

您能使溫度改變的顯示結果稍稍“擴大”，看得更清楚一點嗎？怎麼做？

有數種方式可將顯示"擴大"。假如您已知溫度將僅在200至300K的範圍內變化，則您可修改程式，使螢幕顯示能代表+2至+3伏特的電壓。不過，記得，這樣做並未提高轉換器的解像力。轉換範圍內的電壓步驟數還是一樣不變。您僅擴大了這些值的顯示而已。

您亦可藉用一些其它電路。您可以運算放大器將+2至+3伏特之電壓範圍，放大成0至+5伏特之範圍，致使200至300K之整個溫度範圍即能產生0至+5伏特的電壓。然後這個電壓再加至轉換器且顯示在螢幕上。這樣做由於我們所感興趣的電壓範圍用到了所有256個不同電壓值，故轉換器的解像力亦提高了。

類比轉換器的界面實際還多呢？不過，我們希望以上這些實驗能引起並提高您使用這種元件的興趣。至於進一步的界面觀念與技巧，請您參考" **TRS-80界面實驗，第二册** "以及"**微電腦類比轉換器軟體及硬體界面**"。

請注意，在這個實驗內，我們以兩個1000 歐姆的電阻將+5伏特的電源分成兩半，產生一+2.5伏特的參考電壓。精確的類比轉換器應用均使用+2.500伏特的參考電壓，而不使用電阻。這個實驗我們選用電阻，主要是因為電阻較便宜且易於設定。但是，這樣所產生的結果在精密測量上

就不夠準確。還有其它許多參考元件與電路，在剛剛提到的參考書內都有提到。

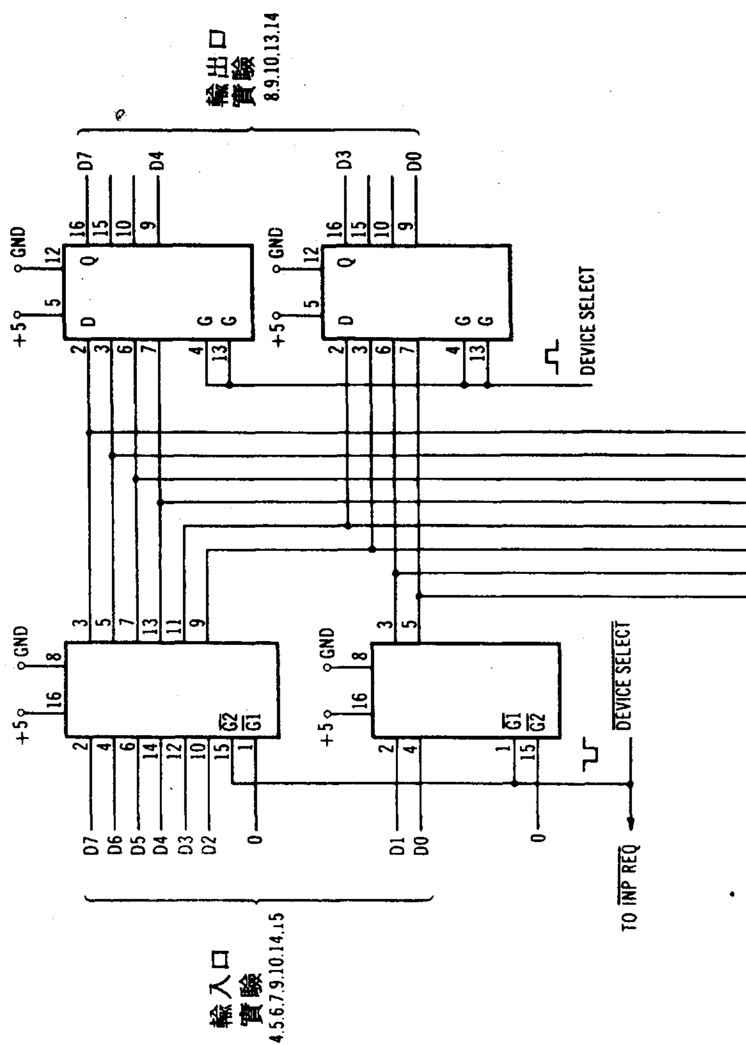
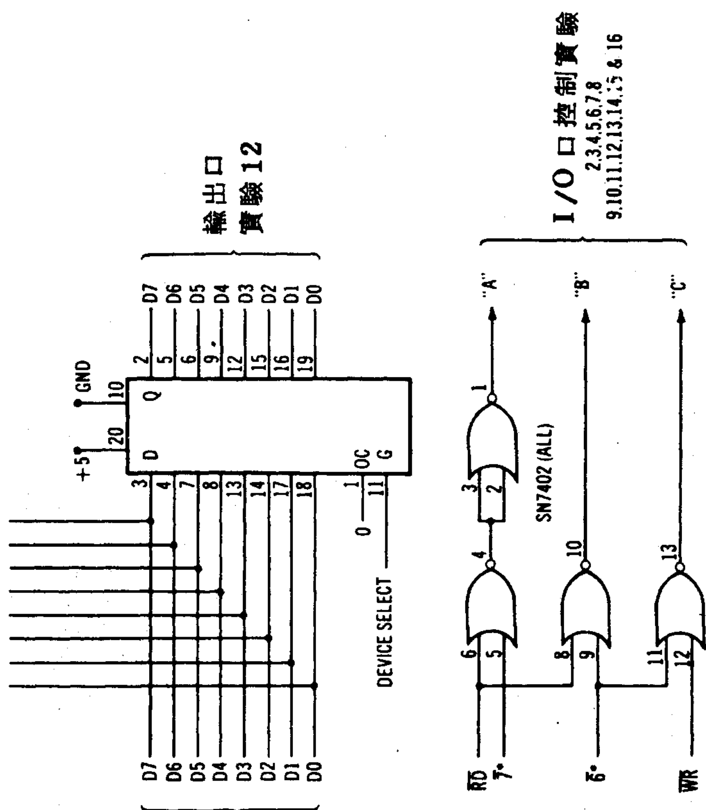


圖 6－27　實驗用之一般化


輸出口
實驗 12
I/O口控制實驗
2,3,4,5,6,7,8
9,10,11,12,13,14,15 & 16
GND
+5
D
Q
OC
G
DEVICE SELECT
SN7402 (ALL)
"A"
"B"
"C"
RD
WR
接至APPLE
之資料巴士
* 解碼器插座位置

I/O口與控制電路圖

7

再談巴士

雖然許多讀者做完第 6 章的所有實驗就覺得夠了，不再做進一步的界面設計與開發，但有些人可能有志於爲自己的計算機系統設計一套永久性的特殊界面電路。本章主要即針對這些讀者所寫。在這一章之內，我們將介紹如何利用 Apple 計算機所提供之特色，設計各種特殊的界面電路。

若您希望設計一套一用再用的界面電路，則您一定不想將之製作於免焊的麵包板上。因爲，麵包板電路佔用空間，看起來雜亂不堪，而且經常在最差的時刻分解了。另一種方式是將界面電路製作成永久形式，使其能置於Apple的殼子內，不受損壞。

Apple 計算機設計時，設計工程師們早已發覺將來用者一定有志於再擴充系統，以使計算機能接上各種標準的週邊以及非標準的電路。因此，Apple計算機在主印刷底板後面均留有八個母邊緣接點，以便幾個重要的信號能輕易爲需要者所用。由於前面所介紹的界面就插入其中一現有的“槽”內，因此，事實上您已用過了部份這些信號。

幾個槽分別由 0 至 7 編號，其中除了第 0 號槽已由製造

商特別留作計算機之特別擴充用外，其餘幾個您都可以用。外面已有許多廠商專賣正好能插入這些槽內的界面，這些界面皆可毫不費力地插入這些槽內。

第 5 章我們曾介紹了一些常用的界面信號：位址巴士信號、資料巴士信號、與一些控制信號。但這七個可用的界面邊界接點上，還有一些其它很有用的信號。這些信號即如表 7－1 所列。

由於資料巴士與位址巴士您都熟透了，因此，這裡我們就不再重複。底下，我們介紹一些其它在界面上非常重要的信號。

7－1　界面控制信號

$\overline{\text{I/O SELECT}}$

表 7－1　Apple巴士信號及說明

接　脚	名　　稱	說　　明
1	$\overline{\text{I/O SELECT}}$	邏輯 0 信號，計算機選取 $C_n00H \sim C_nFFH$ 時，n 號槽的這個信號動作，在 ϕ_0 時動作。0 號槽沒有。（ 10 ）*
2～17	A15 ～ A0	加緩衝之位址巴士線。(5)

18	R / $\overline{W}$	加緩衝之讀/寫控制信號。(2)
19	SYNC	視訊時序同步信號。只有 7 號槽有。（？）
20	$\overline{\text{I/O STROBE}}$	邏輯 0 信號，計算機選取 C800H～CFFFH 時，所有槽的這個信號均動作，於 ϕ_0 時動作。(4)
21	RDY	6502 微處理器之準備好控制輸入。
22	DMA	直接記憶器存取控制輸入。
23	INT OUT	接至鄰接槽之插斷菊環信號。
24	DMA OUT	接至鄰接槽之 DMA 菊環信號。
25	+5伏特	+5伏特電源接點。最高供應各板 500mA。
26	GND	系統接地點。
27	DMA IN	接至鄰接槽之 DMA 菊環信號。
28	INT IN	接至鄰接槽之插斷菊環信號。
29	$\overline{\text{NMI}}$	6502 晶片之不可罩蓋插斷輸入。將處理器引導至 03FBH 處的副程式。

30	$\overline{IRQ}$	6502 晶片之可罩蓋插斷輸入。插斷副程式之位址在 03FE與03FF處。
31	$\overline{RES}$	輸入/輸出線。接邏輯 0 時，Apple重置。界面可監聽或產生重置。
32	$\overline{INH}$	接邏輯 0 時，所有內部的ROM都禁能。
33	－12 V	－12伏特電源接點。總供應電流 200mA 。
4	－5 V	－5 伏特電源接點。總供應電流 200 mA。
35	COLOR REF	這個3.580MHz 彩色參考信號唯有7 號口才有。(?)
36	7 M	標準的7.159MHz 參考信號。(2)
37	Q_3	標準的 2.046MHz 參考信號。(2)
38	ϕ_1	標準的 1.023MHz 微處理器時序信號。(2)
39	USER1	邏輯 0 輸入。接邏輯 0 時，所有內部輸入/輸出設備都禁能。
40	ϕ_0	標準的 1.023MHz 微處

		理器時序信號。ϕ_1 之反相。(2)
41	$\overline{\text{DIVICE SELECT}}$	邏輯 0 信號，各槽一個。各槽均有16個位址使之動作（見表 7 － 3）。(10)
42-49	D7 ～ D0	加緩衝之資料巴士信號。(1)
50	＋ 12V	＋ 12 伏特電源接點。總供應電流 250mA。

由於信號頂上有一橫線，因此，輸入／輸出選取信號，$\overline{\text{I/O SELECT}}$，（第 1 脚）爲邏輯 0 動作。剛剛談到的七個現成界面槽（ 1 號至 7 號）每一個即均具有其各自的輸入／輸出選取信號，因此，這個信號可用以選取任何一塊特定的板。當位址巴士線上之位址爲 C_n00 至 C_nFF（含）之間時，n 號板槽的輸入／輸出選取信號即動作。譬如，若 Apple 之位址巴士上所送的位址爲 C5AB ，則 5 號槽之輸入／輸出選取信號即爲邏輯 0 。這時，其它口之輸入／輸出選取信號即全不會動作。但，有時候這些信號亦全部不會動作。表 7 － 2 所示即爲影響各個輸入／輸出選取信號的位址範圍。

輸入／輸出選取信號有多種可能之用途。由於其在 Apple 計算機選取一段連續的 256 個位置時動作，因此，這個信號可用作一具有 256 個位址之記憶晶片的致能信號。其同樣亦可作爲一可選取 256 個輸入／輸出設備之設備

表 7 － 2　輸入/輸出選取之位址分配

界面槽	位址範圍	
1	C100–C1FF	49408–49663
2	C200–C2FF	49664–49919
3	C300–C3FF	49920–50175
4	C400–C4FF	50176–50431
5	C500–C5FF	50432–50687
6	C600–C6FF	50688–50943
7	C700–C7FF	50944–51199

選取解碼器的致能信號。這兩個應用分別如圖 7 － 1 與 7 － 2 之方塊圖所示。

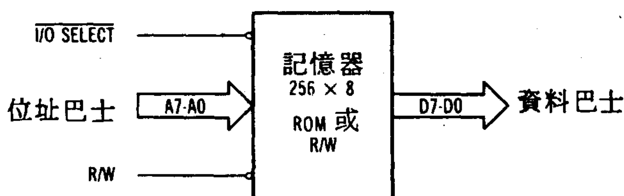


圖 7 － 1　以 $\overline{\text{I/O SELECT}}$ 控制一記憶頁

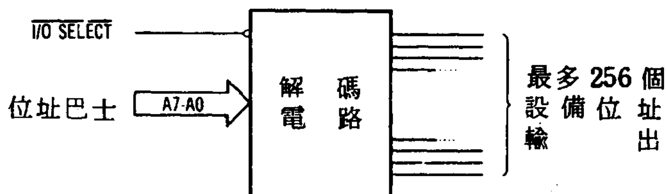


圖 7 － 2　以 $\overline{\text{I/O SELECT}}$ 控制一記憶位址解碼器

您或許要問，既然Apple 本身已能輕易地選取 48K 的

記憶位置了，有人還在Apple計算機系統加上256個位元組的記憶器幹嘛！答案是這樣的，由於組合語言程式效率高，因此，有些應用必須採用一簡短的組合語言常式“推動”界面。在這種情況下，這種“推動常式”（ driver routine ）即可另置於一塊界面電路專用的唯讀記憶（ ROM ）晶片上。如此，推動常式即變成整個界面的一部份，而且當界面板（或卡）一插上，這些常式即自動“取人”（ loaded ），立即可用，而不必像其它常式，再特地由卡帶或磁碟取入。同時，這樣這些常式亦不必佔用任何其它的記憶空間。

另外，有時候界面亦會需要一小量的可讀寫（ R/W ）記憶器作暫時儲存。這時，$\overline{\text{I/O SELECT}}$ 信號亦可用以控制一個單具256個記憶位元組的記憶器。

記得，每一個界面槽均具有其各自的 $\overline{\text{I/O SELECT}}$ 信號，且每一個信號均在 Apple 計算機選取一特定記憶“頁”（ page ）時動作。

$\overline{\text{I/O STROBE}}$

輸入／輸出攫取信號（ $\overline{\text{I/O STROBE}}$ ）是一所有界面槽都有的邏輯0信號。其爲所有界面接點所共用，並不專屬於任一者。每當Apple計算機選取C800H至CFFFH（含）之間的任一記憶位址時，這個信號即爲邏輯0。因此，當位址巴士上的位址落於這個範圍（共含2048＝2K 個記憶位置）時，每一塊板都會收到這個信號。

您可直接以這個信號作記憶晶片與輸入／輸出設備之致

能信號，但您亦可將其與某些位址巴士信號，如 A10～A0，一起再經過一次閘控，以肯定這個信號。圖 7－3 之簡單方塊圖所示即爲這個信號的用法。這個電路以 $\overline{\text{I/O STROBE}}$

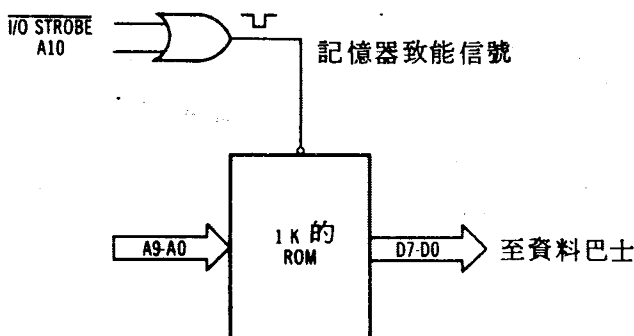


圖 7－3 以 $\overline{\text{I/O STROBE}}$ 信號控制 1K 記憶區段

信號選取界面板上一個1K 的 ROM記憶空間。這個信號另外所能選取的 1K 記憶位址則可分佈於其它界面。不過，作者建議您使用這個信號時還是要多小心，因爲，有時候您會發現，製造商已經用了這條線解碼記憶與輸入／輸出設備位址，而且其用法完全正如以上所述者。因此，您會發現，您不知該選取您希望加至系統之買來界面好，或選取您已自己設計、製作、且裝好的界面好。

$\overline{\text{DEVICE SELECT}}$

這個信號是每個界面槽所特有的，而且，如表 7－3 所示的，每個界面槽僅具有 16 個位址。設備選取（ $\overline{\text{DEVICE}}$

表 7－3　設備選取信號之位址分配

界面槽	位址範圍	
0	C080–C08F	49280–49295
1	C090–C09F	49296–49311
2	C0A0–C0AF	49312–49327
3	C0B0–C0BF	49328–49343
4	C0C0–C0CF	49344–49359
5	C0D0–C0DF	49360–49375
6	C0E0–C0EF	49376–49391
7	C0F0–C0FF	49392–49407

$\overline{\text{SELECT}}$ ）信號亦爲邏輯 0 動作。由於設備選取信號僅動作於一含 16 個位址之區間內，因此，其用途即較偏限於如圖 7－4 所示的輸入／輸出設備選取。圖 7－4 之電路以設備選取信號推動（致能）一 4 線對 16 線之解碼器。若某一界面僅具有一種功能，而且僅需一個致能信號，則您即可選擇單獨使用設備選取信號，而不必再進一步解碼。只要您記

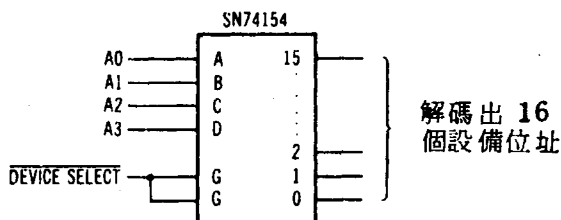


圖 7－4　以 $\overline{\text{DEVICE SELECT}}$ 信號推動一輸出 16 個位址的解碼器

得這樣選取的設備在 16 個不同位址（ C0n0H至 C0nFH ）時均動作，就毫無問題。萬一您決定爾後再作擴充，則這個

信號這樣用法亦限制了您在界面上增加其它功能的能力。

$\overline{\text{IRQ}}$與$\overline{\text{NMI}}$

6502 微處理器晶片上有兩個插斷輸入。$\overline{\text{IRQ}}$（插斷請求）輸入為可罩蓋的（maskable），其可為軟體步驟所禁能。$\overline{\text{NMI}}$（非罩蓋插斷）輸入則恒可得逞（動作）。

這兩條插斷輸入線為全部七個界面槽共用，其中，$\overline{\text{IRQ}}$信號接在 30 脚，$\overline{\text{NMI}}$信號接至 29 脚。於絕大多數界面電路上，$\overline{\text{NMI}}$ 線都特別律定給某一個特定週邊設備，而且無論如何這個輸入信號都要為 6502 微處理器所認可。$\overline{\text{IRQ}}$線則由其它許多界面電路所共用。插斷處理常式內必須含有適當的步驟，使計算機能測知其中那一個設備眞正提出插斷。在這種情況下，每一個插斷設備都必須具有一個位元的輸入口。6502 微處理器只要讀取這個輸入口的資料，即可得知該插斷設備之旗號狀態，知其是否提出插斷。圖 7 － 5 所示即為一典型的插斷旗號電路。留意到，旗號是在軟體程式的

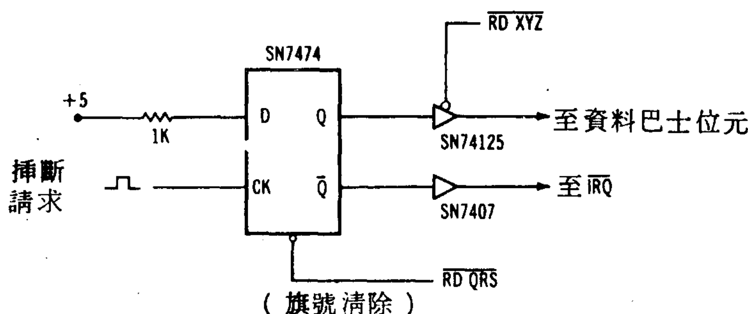


圖 7 － 5　插斷旗號電路圖

控制下清除的。

假如採用這種“取樣式”，或稱“詢問式”，的插斷，教計算機讀取每一設備的狀態，看其是否提出插斷，則軟體程式內即可建立起一套**優先順序**(priority)。舉個例子而言，若程式沿著設備A、B、C……等等的順序讀取各設備的狀態，則設備A的優先順序自然就最高，設備B次之，設備C再次之，如此類推。

幾個界面槽上另外亦有兩個您可能有興趣的插斷線。這兩個信號即爲28脚之插斷輸入(INT IN)以及23 脚的插斷輸出(INT OUT)。其特別用以形成界面板與界面板間之“**菊環**”(daisy chain)插斷結構。如圖7－6所示，信號由一個界面板串接至下一個界面板：1號界面槽的INT OUT信號接至2號界面槽的INT IN，而2號槽的INT OUT信號又接3號槽之INT IN，如此類推。每一

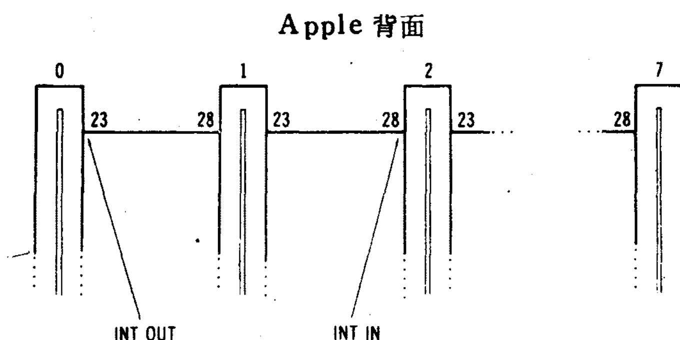


圖7－6　INT IN與INT OUT巴士信號的接法

個界面槽的 INT IN或INT OUT 均僅與隔壁的界面槽連接一次，別不再作其它任何連接。

圖 7 － 7 所示即爲一簡單之菊環插斷結構。愈在“下游”的插斷設備，插斷優先順序即愈低，且離 6502 微處理器之 $\overline{\text{INT}}$ 接脚愈遠。在這個電路內，只要有較高優先之插斷設備提出插斷請求，其它任何較低優先之設備所提出的插斷請求即暫時被忽略。等較高優先之設備被服務完，其插斷旗號清除後，這個較高優先之設備即會將閘門“打開”，讓後面優先順序較低的插斷設備“能”將其插斷請求傳給計算機。

正如您可看出的，計算機還是需要有個辦法得知那一個設備正產生插斷，才能選取適當的插斷服務常式(interrupt service routine)。由於這種插斷方式十分複雜，因此，我們建議您還是採用圖 7 － 5 所示之插斷旗號電路。在絕大多數情況下，這應該夠用。菊環結構內，界面電路板之間不能有“空”槽，否則，整個INT IN/INT OUT電路“環”就會斷掉。插斷至此已討論得差不多了，進一步的細節請您參考“ 6502 程式設計與界面實驗”一書。

$\overline{\text{DMA}}$

$\overline{\text{DMA}}$ 輸入使外部設備能不必經 6502 微處理器而直接選取記憶位置。因此，在這種狀況下，外部設備直接存取了記憶器(Direct Memory Access，簡稱DMA)。由於有好幾個設備均可請求直接存取記憶器，而幾個界面槽又都有可如INT IN與INT OUT連接的DMA IN與DMA

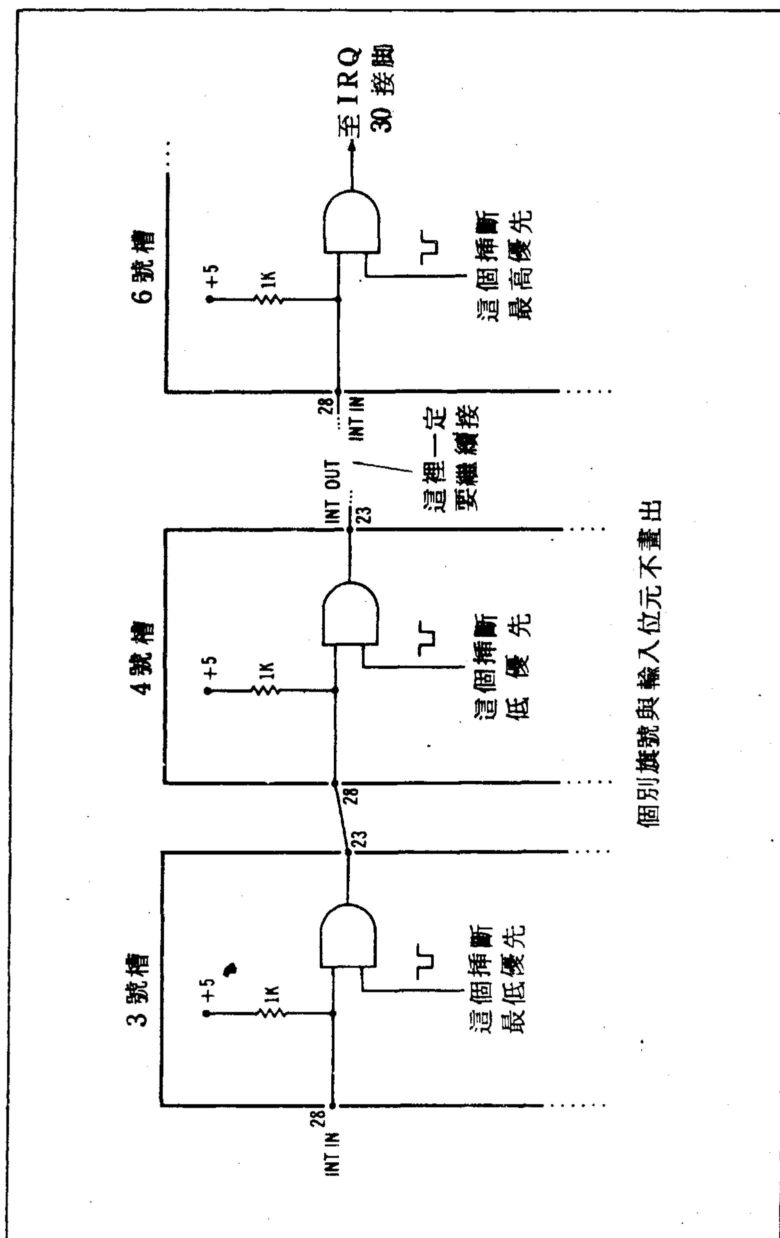

3號槽
4號槽
6號槽
+5
1K
INT IN
28
23
INT OUT
這個插斷最低優先
這個插斷優先
這個插斷最高優先
這裡一定要繼續接
至IRQ 30接腳
個別旗號與輸入位元不畫出

圖7－7　菊環插斷結構

OUT 接脚，因此，各週邊設備之間亦可形成一菊環結構。由於直接存取界面設計並非一件十分簡易的工作，因此，在使用這個特色前，我們建議您能先徹底地了解 6502 微處理器與其有關電路之動作原理。

$\overline{\text{RES}}$

第 31 脚之重置信號，$\overline{\text{RES}}$，實際上是一雙向的信號線。由於每當電源一加上 Apple 重置時，或您按 RESET 按鈕時，這個信號均爲邏輯零，所以，您可以這條線重置您的界面電路。只要將這條線接地，您亦可迫使 Apple 變成重置狀態。若您選擇以這條線自界面電路重置 Apple，則您必須以一高電流之集極開路閘或緩衝器，將這條線接地。SN7407 集極開路緩衝器晶片這時就是個適當人選。$\overline{\text{RES}}$ 信號線爲所有界面槽共用。

$\overline{\text{INH}}$

於 Apple 計算機內，您可以自己所寫的組合語言程式取代儲存於 BASIC 解釋程式 ROM 內的程式。只要將第 32 脚之 $\overline{\text{INH}}$ 線接地，您即可使儲存 BASIC 解釋程式以及監督程式的所有 ROM 都失效，進而以您自己所寫的程式控制整個系統。由於這種作業已早留有餘地，所以，您很可能用不到此一功能，因爲，這樣您就無法存取 Apple 所供應之標準 ROM 內的所有副程式。舉個例子而言，沒有 BASIC 解釋

程式 ROM 內的副程式，您就很難控制顯示幕。但假如您想使用這個功能，則您就需以一集極開路緩衝器晶片將這條線接地。

USER 1

這個輸入使您能教Apple計算機停止產生 I/O SELECT 與DEVICE SELECT 信號，以致能“關掉”所有的輸入/輸出設備。欲產生此一作業時，這條線必須接邏輯 0 。爲了防止不小心用錯了這條線，因此，欲使用 USER 1 信號時，您必須以一條跳線將Apple主印刷底板上的兩個焊接點接起，您才有辦法使用這個信號。有關這個信號使用的詳細細節，請您參考“ Apple Ⅱ參考手册”。

由於使用 I/O SELECT與DEVICE SELECT 信號的主要目的即在簡化界面設計，因此，除非您想將系統擴充，接上一些基本系統以外，或可能用到已分配給 I/O SELECT與DEVICE SELECT信號的記憶位址時，您才有可能用到這條線。 USER 1 信號在界面接點上的第 39 脚。

RDY

有時候，我們必須令 6502 微處理器稍爲“延遲”一下，致使輸入/輸出設備或記憶器晶片有足夠的時間讀取資料，並將資料置於資料巴士上。只要將每一界面接點 21 脚上的準備好輸入 (RDY) 接地，我們即可令 6502 微處理器處

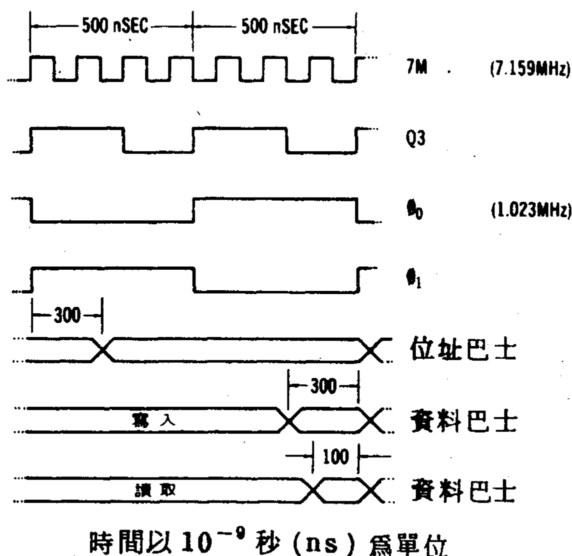
於"等候"狀態。這個輸入必須與微處理器的時序取得同步，而且其應於 ϕ_1 時序爲邏輯 1 準位時轉態。由於較早期記憶元件之資料存取速度均無法達到計算機之要求，因此，RDY 輸入均用於較早期的 6502 系統。在這些系統內，只要送出位址資訊，6502 微處理器即必須進至"等候"狀態數個時序週期，一直等到資料備好於資料巴士上爲止。除了很特殊的界面外，相信您很少會用到這個信號。

時序信號

Apple 上有六個時序信號可作界面用。這些信號分別是 ϕ_0、ϕ_1、Q_3、7M、COLOR、REF、與 SYNC。ϕ_0 與 ϕ_1 爲主時序信號，頻率 1MHz 。兩個時序信號彼此反相。這些信號主要用以控制外部輸入/輸出作業，使資料於巴士上能正常流通。如圖 5－12 所示，ϕ_1 信號用以控制外部輸入/輸出設備的 $\overline{RD}$ 與 $\overline{WR}$ 信號產生。輸入/輸出接點處的 $\overline{I/O\ SELECT}$ 與 $\overline{DEVICE\ SELECT}$ 信信號都已事先與 ϕ_1 時序信號"閘控"認定過。

Q_3 信號則爲一頻率 2MHz 的非對稱時序信號；亦即，其並非方波。7M 信號則爲一波形爲方波，頻率爲 7MHz 的時序信號。幾個時序信號均由 Apple 內的主時序電路得來，其彼此間的關係如圖 7 － 8 所示。有關 6502 之時序關係，請讀者進一步參考 6502 微處理器之數據說明書。

COLOR REF 與 SYNC 信號唯有 7 號界面槽有。COLOR REF 信號爲 Apple 之視頻時序電路所產生的彩色



時間以 10^{-9} 秒 (ns) 爲單位

圖 7－8　Apple之各種時序信號的時序圖

參考信號，其頻率爲 3.5 MHz 。除非您用到視頻控制電路，否則，您將不可能用到這些信號。

電　源

界面接點提供了四種標準電壓與接地。這些電壓分別爲 +12 、−12 、+5 、與−5 伏特。由於每一電壓的電流均僅有幾百毫安 (milliamperes)，所以，您應考慮採用諸如 SN74LS00 族中的低功率界面晶片。

其它考慮事項

界面信號的巴士推動能力均非常有限，絕大多數信號均僅侷限於推動少數幾個SN74LS00 型的輸入。因此，您在設計時必須非常小心，切勿使這些信號因推動太多的晶片輸入而發生過載現象。倘若這些信號所必須推動的晶片輸入超出其能力範圍內，則這時信號就必須先經緩衝器晶片緩衝。記得，緩衝器本身亦需由電源取用一部份功率，因此，界面接點事實上亦未增加太多的“額外”功率。是以，信號緩衝所需必須與現有功率取得平衡。當然，您亦可另外使用一套電源，專門供應所有界面板的電力，但這就失去了將界面電路置於Apple 封匣內的最初目的了。

7－2　界面例題

現在，絕大多數有用的界面信號都已介紹完了，接著讓我們來仔細研究一個能用於Apple計算機的典型界面電路。在許多應用上，計算機都必須與其它設備溝通。這些設備可能包括印表機（printers）、控制器、遙端資料詢問站、或甚至其它計算機。大多數情況下，計算機與這些設備間的溝通都會採用一種稱爲串行溝通（serial communication）的形式，以避免使用多導線的長電纜。絕大多數串行溝通方式均使用3或4條電線，因此，欲交換的資訊只得以串行的

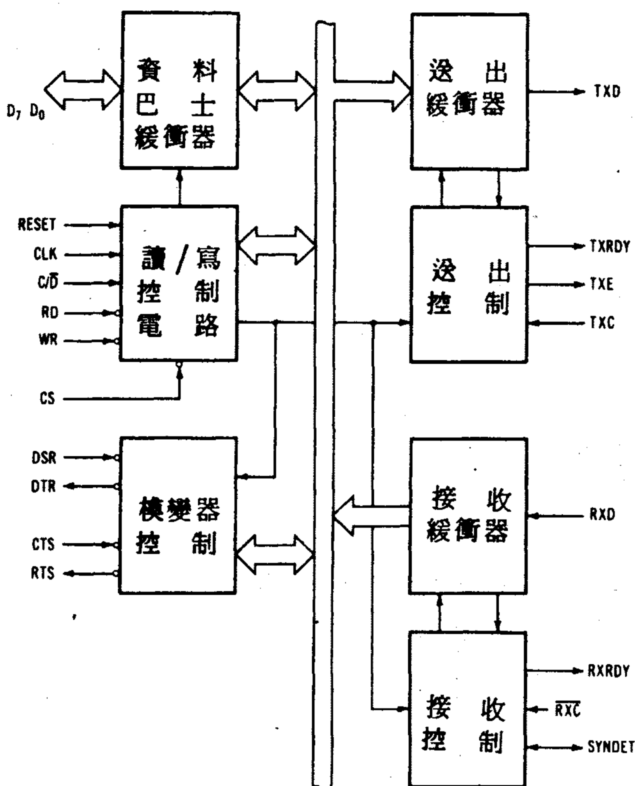
方式，一個位元接一個位元地經電線傳遞。當然，某一設備送出的訊號專用一組電線，而接收的信號又專用另一組電線。由於兩個溝通的系統間無共用的時序信號或參考點連接，因此，這種通訊通常稱為“**非同步串行通訊**”。

在設計微處理器晶片的同時，絕大多數微處理器製造商都專門設計有適合其微處理器用的各種溝通晶片。事實上，這種東西亦可“跨”族使用；亦即，針對8080A 所設計的溝通晶片亦可用於 6502 微處理器上。而事實上，這正是這個例子我們所欲詮釋的；將一8251 萬用同步 / 非同步接收 / 送出器（universal sychronous/asychronous receiver /transmitter 簡寫為 **USART** ）晶片界面至Apple計算機上。但這兒我們並不詳細地討論USART晶片的動作原理，有關這部份，請讀者自行參考“**TRS-80 界面，第 2 册**”。

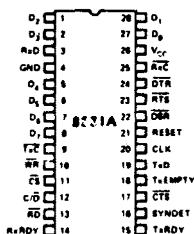
8251 USART由於是一巴士吻合（bus-compatible）晶片，因此，應不難接至Apple上。圖7－9所示即為USART晶片之接腳圖與方塊圖。從這圖中，您應能自行找出資料巴士輸入，$\overline{RD}$與$\overline{WR}$控制輸入以及晶片選擇輸入$\overline{CS}$。USART由於含有兩組暫存器，因此，必須有辦法加以區別。晶片上第12脚之CONTROL/$\overline{DATA}$（C/$\overline{D}$）輸入的功用即在此。這條線上邏輯1時選取控制型態或命令型態，邏輯0時選取資料型態。您可將一個位址位元接至這個輸入上，以使計算機能以某一個位址選取控制暫存器，而以另一位址選取資料暫存器。

由於USART將與其它非同步串行元件溝通，因此，我們必須使用標準的資料傳輸速度，以確保送出資料的儀器與

方塊圖



接 脚 圖



Pin Name	Pin Function
D_7 D_0	Data Bus (8 bits)
C/D	Control or Data is to be Written or Read
$\overline{RD}$	Read Data Command
$\overline{WR}$	Write Data or Control Command
$\overline{CS}$	Chip Enable
CLK	Clock Pulse (TTL)
RESET	Reset
$\overline{TxC}$	Transmitter Clock
TxD	Transmitter Data
$\overline{RxC}$	Receiver Clock
RxD	Receiver Data
RxRDY	Receiver Ready (has character for 8080)
TxRDY	Transmitter Ready (ready for char. from 8080)

Pin Name	Pin Function
$\overline{DSR}$	Data Set Ready
$\overline{DTR}$	Data Terminal Ready
SYNDET	Sync Detect
$\overline{RTS}$	Request to Send Data
$\overline{CTS}$	Clear to Send Data
TxE	Transmitter Empty
Vcc	+5 Volt Supply
GND	Ground

圖 7－9　8251A USART 晶片之接脚及方塊圖

接收資料之計算機兩者的資料傳輸速度相當接近。由於是晶體控制的，因此，我們選擇了Motorola MC14411 位元率產生晶片擔任此一角色。當然，還有其它很常用的時序產生方式。

由於SN7400 TTL 族元件所產生的標準邏輯準位無法用以推動長通信線，因此，您必須決定是選用 20mA之電流迴路信號或標準的RS-232C控制準位。這些必要的準位轉換電路極易獲得，其在前面所提過的參考書內均有詳細的說明。

除非有軟體配合，否則，任何界面晶片形如空架。USART晶片亦然。由於簡單，因此，絕大部情況下您都希望採用 BASIC 語言程式。萬一您選用組合語言程式，則您最好將您所寫的控制程式置於 ROM 內，並將 ROM 置於界面板上。由於每一界面槽均有 256 個位元組的記憶空間可用，因此，其可容納一個小小的 ROM 。就某些USART 控制程式而言，256 個位元組的記憶空間已足足有餘了。在置於 ROM 內前，您可先以監督程式測試一下您自己所設計的組合語言程式。

圖7－10所示即爲一完整的USART界面。這個電路已在Apple計算機上接好並測試過了。 若您希望將這個電路用在您的計算機上，則您最好先找來 8251 或 8251A USART 晶片以及Motorola MC14411 位元率產生晶片的資料說明書，先研究一下這些晶片的動作原理。圖 7－11 所示即爲一可用以儲存USART 組合語言控制程式之 256位元組 ROM 的一般性選取電路。實際的電路視您所選用的 ROM 晶片而

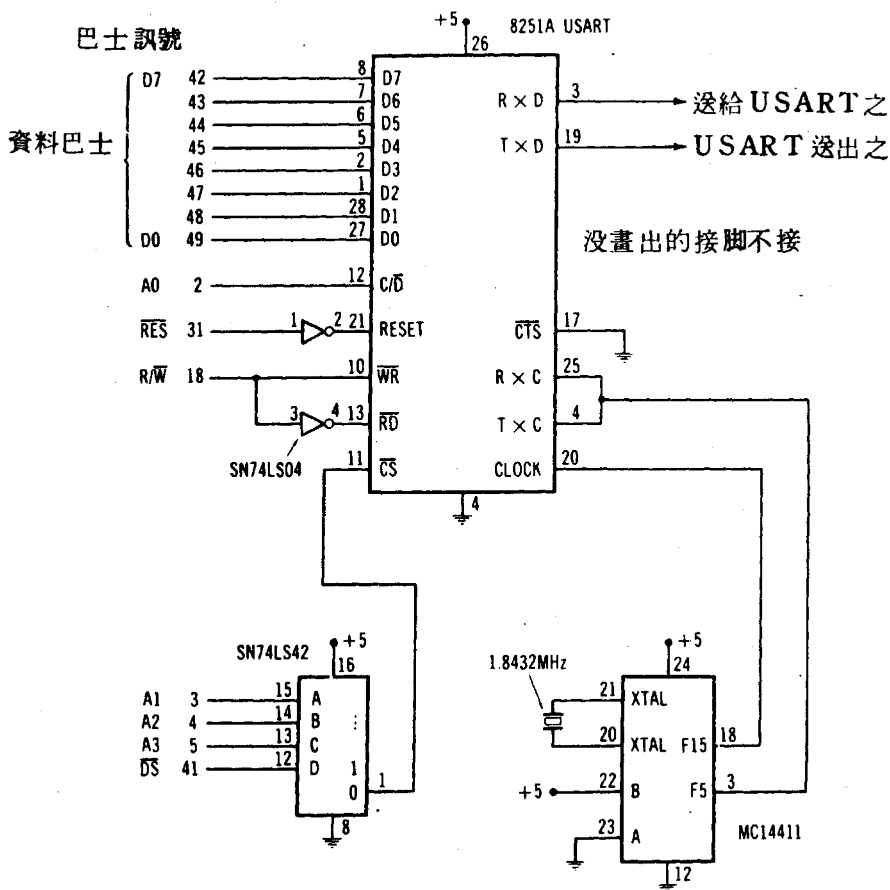


圖 7－10　USART至Apple之簡易界面電路圖

稍有差異。在這個電路內，我們使用了兩個Fairchild 93427 ROM。這是一種快速、雙極子、可熔性連接的ROM。每一晶片各含 1024 位元，組織成 256 個 4 位元的字組。因此，組合成 8 位元的字組正好需要 2 個晶片。在此建議您最好勿採用速度慢又可抹拭的 PROM ，因爲，這種晶片的存取速

度相當慢，可能造成問題。這些元件的記憶容量通常亦超出您所需者。

您可將這個電路製作在一標準的界面包線板上，亦可將之製作於可插入現有界面槽之合適模型板上。若您採用包線模型技巧，則您會發現，包線接脚與晶片在板的兩側均突出，裝上去後，使得鄰接的界面槽很難使用。

在我們的例子內，我們採用 3 號槽作 USART 界面，因此，USART 的位址是49328至49329。位址49328所選取之暫存器爲接收與送出暫存器，而位址49329所選取之暫存器則爲控制與旗號暫存器。記得，每一個位址均選取兩個暫存器，其中一個爲欲寫入之暫存器，另一個爲欲讀取之暫存器。如表 7 － 3 所示的，當界面板移至另一個界面槽時，USART的位址即變更。

欲使用 USART 界面時，首先您必須連續送兩個控制位

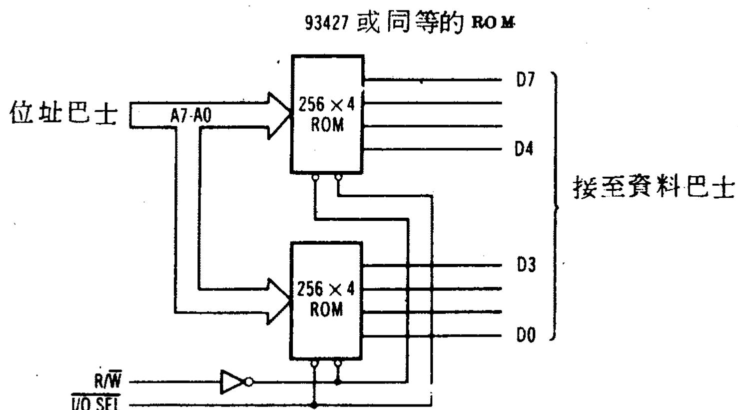


圖 7 － 11　256 位元組記憶器之擴充電路圖

元組至控制暫存器，將晶片啓動。您不必擔心爲何兩個位元組均送至同一個暫存器，USART 自己會“知道”如何處置的。USART 經啓動之後，其即可開始送出與接收非同步的串行資料串。例題 7 － 1 所示即爲一可用以送出八位元資料的程式，而例題 7 － 2 所示則爲可用以接收一資料位元組的程式。

例題 7 － 1　USART 送出控制副程式

```
1010  POKE 49328, TX
1020  WAIT 49329, 1
1030  RETURN
```

例題 7 － 2　USART 接收控制副程式

```
1050  WAIT 49329, 2
1060  RX = PEEK(49328)
1070  RETURN
```

兩個程式均檢查必要的旗號，以確保送出器於準備好時才送出資料，且接收器僅有在實際收到資料後才提供資料。

我們舉這個例題的主要目的是，藉設計一個用到許多 Apple 巴士界面控制信號的簡單界面，使您能知道這些信號如何動作。由於可用於實際應用上，因此，若您能同時了解這個界面例題的實際動作情形那是最好了。我們希望您能由此一例題看出設計 Apple 界面是何等的容易；說穿了就是我們一直所介紹的口選取、口控制、與旗號罷了。

8

馬達、燈泡、電鈴與汽笛

在從事微電腦界面工作時，人們隨時都會面臨以計算機控制實際設備的問題，同時亦需控制，或推動，需高電壓與／或高電流的設備。不論您欲以計算機控制幾個ＬＥＤ，或一個高功率馬達，您都必須想到能輕易將計算機所輸出之邏輯信號，轉換成能適合於受控制之外部設備所需的電壓與電流。這一章，我們將介紹一些由取用低電壓低電流，至高電壓高電流的設備推動電路（或稱設備控制器），當然，任何一件界面工作隨時都可能有另外的方法可達成，因此，這章我們僅選擇性地介紹一些您將認爲很適合您界面所需的典型電路，而不將所有各種不同的推動電路一一介紹。由於控制這些元件的軟體都屬比較次要的東西，因此，這章我們主要將介紹這些界面電路的硬體方面。

本章目標

讀完這一章之後，您將會：

- 設計以集極開路閘與解碼器推動低電壓與低電流設備的電路。
- 說出集極開路式解碼器在記憶器擴接解碼上的用途。
- 說出現有的各種不同週邊推動器與電晶體陣列。
- 設計高電流的燈炮推動電路。
- 討論燈炮推動電路之波動電流。
- 設計使用電晶體陣列的推動電路。
- 設計可選取的鎖住推動器電路。
- 設計控制馬達、燈泡、與其他線電壓動作設備之固態繼電器電路。
- 討論隔離用之光交連的用法。

8－1 集極開路式電路

最簡單的推動器電路永遠屬於 SN7400 系列 IC 。這些元件之輸出電晶體的集極（ collector）未接，作爲某一特定邏輯閘、解碼器、多工器、或其他**集極開路式**（ open-coll-ecter-type ）功能的輸出。典型的集極開路式元件即如圖 8 - 1 之電路所示。

這個電路，正好爲一反相器，內，輸出端就是輸出電晶體的未接集極。注意，但這個電晶體的射極接地。這意思說，在導電時，這個電晶體的輸出將等於接地電位。不過，在不導電時，這個未接電晶體的集極將沒電壓。因此與其他 SN7400 系件元件不同的是，集極開路式元件無邏輯 1 輸出

。其他元件由於輸出電晶體之集極都有接，因此，當輸出電晶體不導電時，輸出均被“提升”至邏輯 1 準位。提升功能另外使用其他的內部電路。記得，絕大多數 SN7400 系列元件都有邏輯 1 或邏輯 0 輸出狀態，但集極開路式元件僅提供一接地路徑，無任何提升動作。如此一來，您可能要問，那爲什麼人們對這種集極開路式元件這麼有興趣呢？

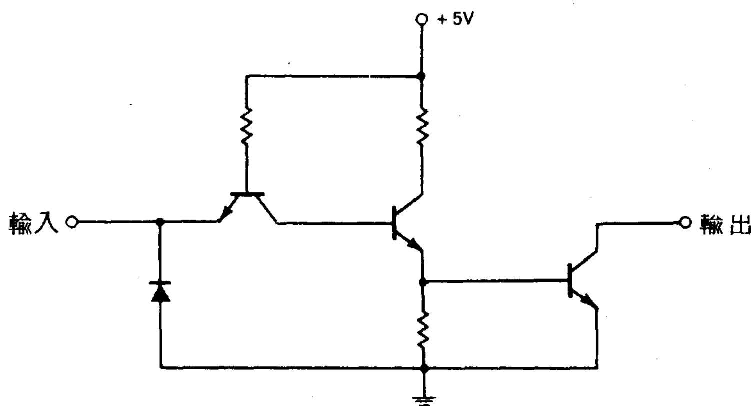


圖 8－1　SN7405集極開路式反相器的電路圖

早在三態元件被廣泛採用之前，集極開路式元件在計算機電路上就已十分重要了，因爲，集極開路式可宛如與信號線“斷接”一般。據此，其可將一信號線接地，或與信號線斷接，使其它元件能控制此一信號線。假若這條巴士上的元件輸出不同的邏輯 1 ，則巴士上同時出現邏輯 1 與邏輯 0 狀態將導致某些元件被燒毀。由於諸集極開路元件中無任一者可產生不同之邏輯 1 輸出，故我們以一簡單之提升電阻，使巴士成邏輯 1 狀態，除非有其中之一集極開路閘藉其輸出電

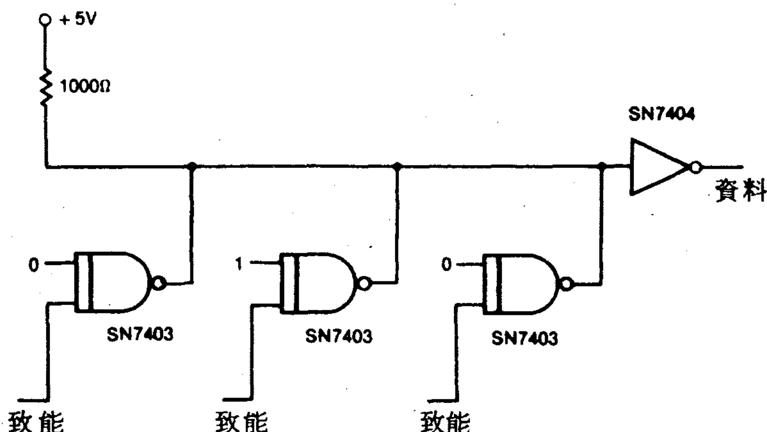


圖 8－2　典型的集極開路式巴士結構

晶體將巴士拉至地電位。許多早期的迷你計算機，如DEC公司之 PDP—8 系列，均使用這種集極開路式巴士技巧。圖 8－2 所示即爲一典型之集極開路式巴士。這個電路每次令一個不同的巴士推動器連接至巴士上，使呈現於推動器輸入端之資料能傳遞至巴士上，以及至接收設備（在例中爲一簡單反相器）上。爲了區分集極開路式邏輯閘與一般正常之邏輯閘，我們特將集極開路式元件之輸入邊畫成雙線。

於三態巴士系統內，每一瞬間僅能有一個輸出至巴士的輸出動作。但於集極開路式系統內，這就不成問題，因爲，電流永遠受巴士尾端之 1000歐姆的提升電阻所限制，致兩個或兩個以上的巴士推動器動作並不致增加巴士上的電流。在圖 8－2 之電路內，您應發現，資料傳經集極開路式推動器時被反相一次，邏輯 0 變邏輯 1 ，邏輯 1 變邏輯 0 。然後，接收的反相器電路再將資料反相一次。巴士之平常備用狀態

爲邏輯 1。

雖然集極開路式元件目前在計算機巴士與界面上仍很常用，但其已逐漸爲三態巴士與三態巴士推動器所取代。雖然三態巴士較優，但由於能構成或打斷接地的路徑，因此，集極開路式元件目前仍很流行。是以，其可輕易操作推動諸如小簧片繼電器與LED 等低電流與低電壓元件之用。事實上，集極開路式元件是“取用”(sink) 到地的電流，但這一類的元件平常還是稱“推動器”(driver)。若您希望推動的是幾個低電壓、低電流元件，您會發現，集極開路式元件是最理想不過了！

表 8－1　一些SN74系列元件之電流取用能力

元件	族				
	74－	74H－	74L－	74LS－	74S－
－00 Quad NAND	16	20	3.6	8	20
－03 Quad NAND	16	20	3.6	8	20
－05 Hex INVERTER	16	20	3.6	8	20
－12 Tri NAND	16	20	3.6	8	20
－22 Dual NAND	16	20	3.6	8	20

附註：所有電流均爲mA，電壓最高 5.5 伏特。

表 8 － 1 所示即爲 SN7400 系列元件中，一些集極開路式元件的電流取用能力。取用電流最大者乃 SN 74H 系列以及 SN74S 系列，每一輸出爲 20mA。這個數值極適合於小簧片繼電器以及 LED。雖然標準的 SN7400 系列元件之電流取用能力爲 16mA，但由於其具有產生邏輯 1 輸出之活性 (active) 提升元件，故其不應用以推動繼電器 LED，或其它的非邏輯元件。同樣地，若不用集極開路式推動器，您亦不應以一鎖住器，正反器，或計數器的輸出推動一非邏輯性元件。圖 8 － 3 所示即爲以集極開路式元件控制LED顯示器以

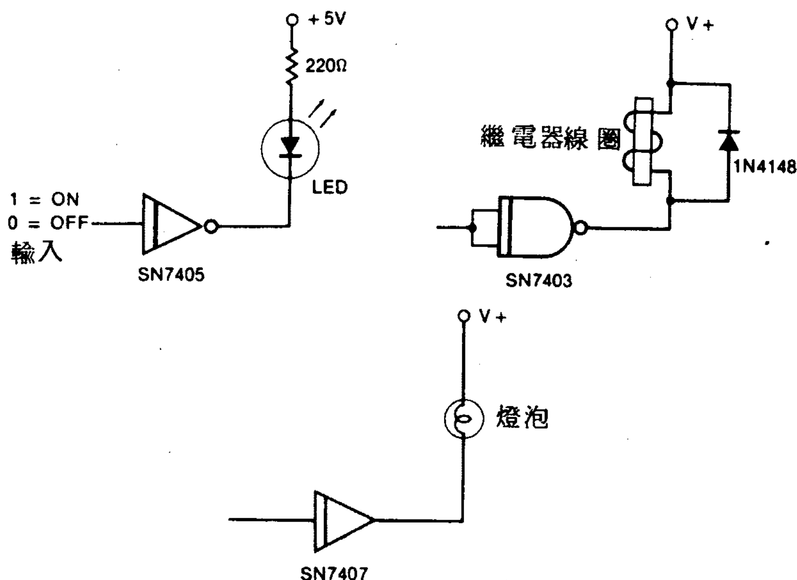


圖 8 — 3　集極開路式推動器之典型用法

及低電流繼電器電路之典型應用。表 8 — 1 之附註您該有注意到，集極開路式元件可加之最大電壓為+5.5 伏特。

不過，某些應用却需要有高電壓與高電流的集極開路式元件。SN7400 系列內另外有一部份元件是屬於這一類的。這些集極開路式元件正如表 8 — 2 所列者。注意這些元件全

表 8 — 2　若干集極開路式元件之最大電流與電壓準位

元　　件	I_{MAX} (mA)	V_{MAX} (V)
7406 Hex INVERTER	40	30
7407 Hex BUFFER	40	30
7416 Hex INVERTER	40	15
7417 Hex BUFFER	40	15
7426 Quad 2-input NAND	16	15
7433 Quad 2-input NOR	48	5.5
7438 Quad 2-input NAND	48	5.5

部是屬於SN7400 系列，而非SN74LS-，SN74S-或其它的族。這些緩衝器、反相器、與邏輯閘全部具有高電壓與高電流的能力。這幾類型的元件經常用於計算機必須推動一些高電壓負載，譬如，推動一十 12 伏特繼電器的場合。高電壓集極開路式元件亦用作電晶體一電晶體邏輯（TTL）族與其它以高電壓代表邏輯 1 與邏輯 0 準位之邏輯族之間的界面。譬如，作爲一TTL電路以及一使用CMOS邏輯晶片之電路間的界面就是壹個例子。圖 8 — 4 所示即爲一樣本電路，若外部數位電路爲CMOS元件，則這種界面技術就十分重要。由於雜音免役性優於TTL族，故CMOS元件經常用於工業應用上，且由於其功率損耗較低，故其亦廣泛應用於遙端或以電池作電源供給的應用上。

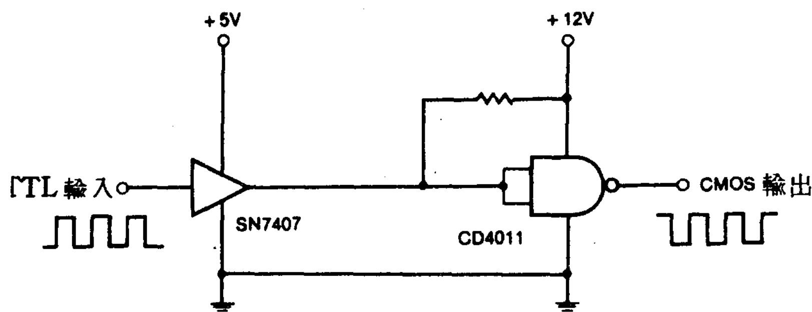


圖 8 — 4　以集極開路式緩衝器作爲TTL至CMOS之界面元件

於圖 8 — 3 之界面內，繼電器之線圈兩端接有一二極體。由於電路加上電源後，這個二極體根本沒有用途，因此，您或許要問，加這個二極體幹什麼？不知您曾否記得，當線

圈兩端所加之電源拿掉時，“崩潰”的電流會產生一反電動勢（emf），產生一個電壓。這個電壓在大部份繼電器上都相當大。只要將您的手指置於一動作中之6伏特的蜂鳴器兩端，您即可感覺出這個電壓有多大。

爲了防止這個高電壓脈衝破壞了繼電器推動電路中之集極開路式推動器晶片，因此，我們加了一個二極體傍路這個反電動勢。只要用到繼電器，即記得加上這麼一個傍路或保護二極體。

同樣地，雖然目前的趨勢已逐漸以LED 作面板指示燈或守夜燈等等，但如圖8－3所示的，集極開路式推動器亦可推動白熾燈泡。若您選用這種白熾燈泡，您即應記得，當燈絲冷却時，這種燈泡有一股很大的“入侵”（inrush）電流。這個電流可能出現數十毫秒之久，且經常超過推動燈泡之推動器的最大額定電流。這點在後面我們討論推動需117伏特交流線電壓時，會有更詳細的解說。籠統來說，爲了克服入侵電流，白熾燈泡都需要有較高額定電流的推動器。

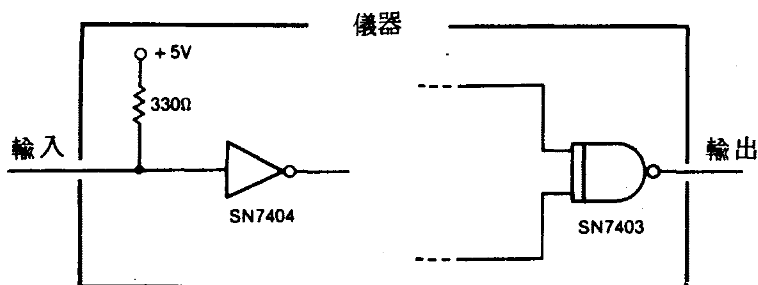


圖8－5 儀器之典型輸入與輸出

若您所要接的是在商場上買得到的電子元件及儀器，那您一定經常會碰到如圖 8－5 所示，具有提升電阻，又與 TTL相吻合的輸入。同時，您也會碰到如圖 8－5所示，由集極開路式元件所推動的輸出。雖然標準的TTL輸出或許可以推動具有提升電阻的輸入，但最好還是不要用。事實上，由於這種儀器絕大部份都與計算機或界面有一段距離，因此，集極開路式元件乃是最佳的人選。圖 8－6 所示即爲其中一典型例子，採用這種集極開路式推動器的理由之一，乃因其能推動具有相當大電流的長線，使信號較不容易受雜音感應的影響。集極開路式元件同時亦可以減低信號導體之線電阻的影響。就圖 8－6 之例子而言，在送出邏輯 0 於每一條信號線時，每一集極開路式推動器將取用 11 mA 的電流。

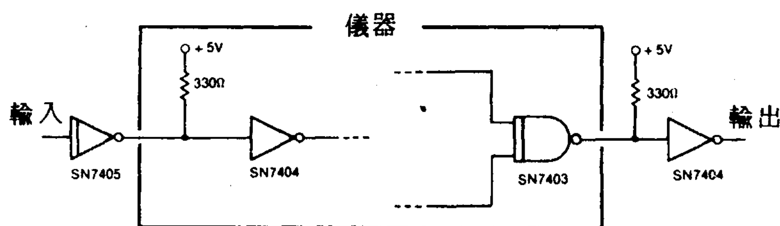
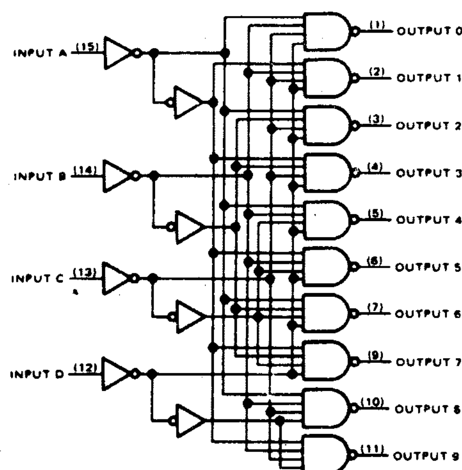


圖 8－6　以集極開路式元件作標準儀器輸入/輸出線的界面

8－2　集極開路式解碼器

雖然集極開路式邏輯閘很好用，但有些場合還是需要其

它的集極開路式功能。其中，最常見的就是集極開路式解碼器。在標準的 SN7400 族內，有三個解碼器值得一提：SN74145 高電流十進解碼器、SN74159四線對十六線解碼器，以及 SN74141高電壓十進解碼器。



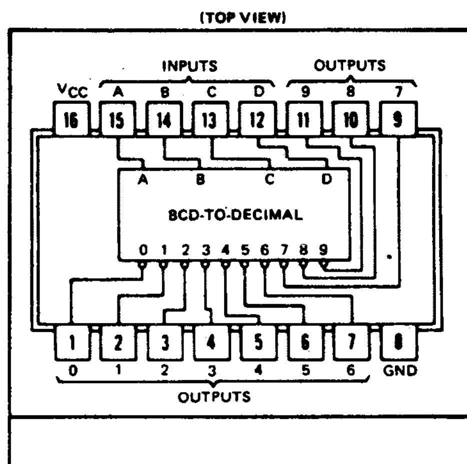


圖 8－7　SN 74145解碼器晶片之方塊圖與接腳圖

圖 8 — 7 所示即 SN 74145十進解碼器之方塊圖與接腳圖。這個解碼器接收四位元（四條線）的二進輸入訊息，將之解碼，並使10個輸出其中之一導電。由於僅有 0 至 9 等10個輸出，因此，並非四位元輸入的所有組合都被解碼。事實上，只有 0000至1001的二進輸入才算有效的輸入，其餘的所有輸入則都將使輸出電晶體不導電。由於每一個輸出電晶體最高均可在15伏特之電壓下取用 80 mA 的電流，所以，這個晶片在推動LED或白熾燈泡顯示上特別有用。由此可見，解碼器亦可用以產生脈衝，推動繼電器與其它元件。

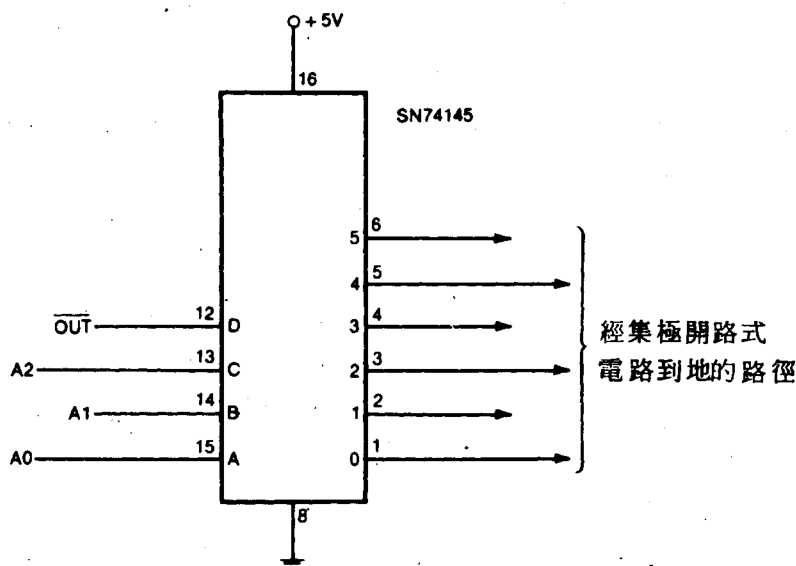


圖 8 — 8　以SN 74145解碼器產生控制用的短暫電流取用脈衝

為了推動外部儀器與電路，計算機經常亦須產生高電流

脈衝。這種情況下，我們可以一集極開路式邏輯閘或反相器與一標準設備位址解碼器併用，以產生這些脈衝；或另外特用一 SN74145 解碼器，在一塊 I C 內同時產生解碼與推動兩種功能。圖 8 — 8 之電路圖所示，即如何以這種集極開路式解碼器提供外部設備使用之對地短暫高電流流徑的情形。注意，這種電路假設所需之提升或電壓源元件設在解碼器所接的界面內。圖 8 — 8 所示的設備位址解碼方式是屬於非絕對的位址解碼，因此，同時會有多個不同之設備位址會產生高電流脈衝。

圖 8 — 9 所示乃集極開路式元件的另一種用法。圖中之電路將數個集極開路式元件的輸出連在一起，一同接至一提升電阻上。由此您可看出，標準 SN7400 系列的元件輸出彼此無法接在一起，但集極開路式元件則能。圖 8 — 9 之電路以集極開路式元件組成一 NAND閘功能。 SN74145 解碼器的 D 輸入唯有在緩衝器與反相器之輸入呈現適當的邏輯 0 與邏輯 1 組合時，才會呈邏輯 0 。此即所謂的NAND函數。因爲，邏輯 1 只是被後衝而已，而邏輯 0 則被反相成邏輯 1 。當所有輸入均爲邏輯 1 時，NAND 閘即出現獨一無二的邏輯 0 狀態。當 SN74145 解碼器的D 輸入爲邏輯 0 時，解碼器即致能，解碼器輸出端之適當集極開路式輸出導電，使電流流入接地端。在使用輸出述句之軟體的控制下，圖 8 — 9 所示的電路能產生八個脈衝。您應還記得，在執行這種述句時，設備位址出現於位址巴士上，而資料值則出現於資料巴士上。

在這個例子內，我們曾說， SN74145 解碼器唯有在其D 輸入爲邏輯 0 時才致能，事實上並非如此，因爲，解碼器永

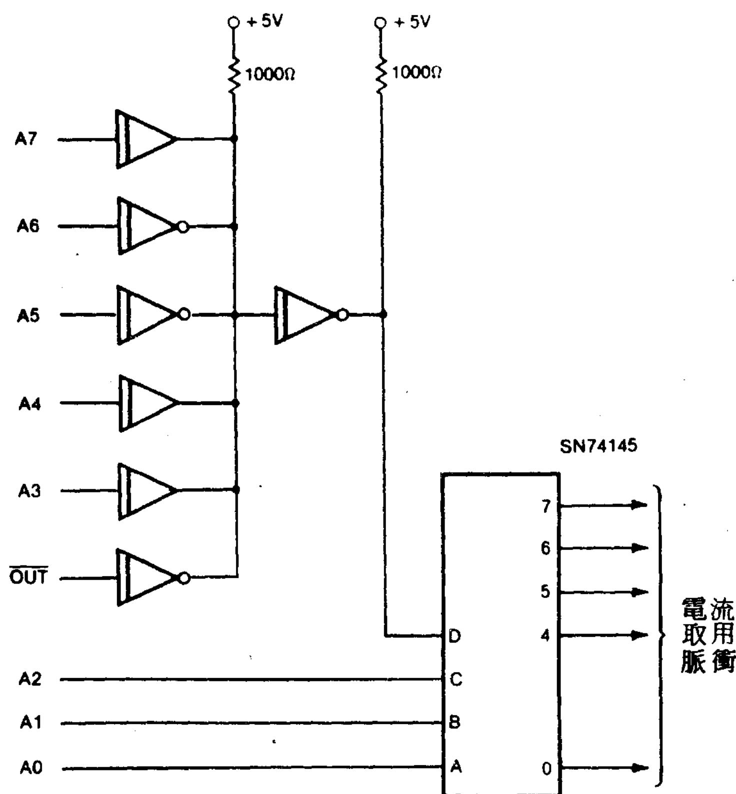


圖 8 — 9　絕對解碼之電流取用脈衝產生電路

遠接受與解碼四個二進位元。不過，若您忘了使用解碼器的"8"與"9"輸出，您就會認爲D輸入是一致能輸入，因爲，唯有當D輸入爲邏輯0時，解碼器才能使所餘之"0"至"7"其中之一輸出動作。

在圖 8 — 8 與 8 — 9 所舉例，以 SN74145 爲主的解碼方

法中，解碼器的功用均爲產生短暫脈衝。若您希望集極開路式輸出被致能的時間超過輸出指令脈衝的長度，則您就必須使用鎖住器。在這種情況下，資料值先送出至鎖住器，然後

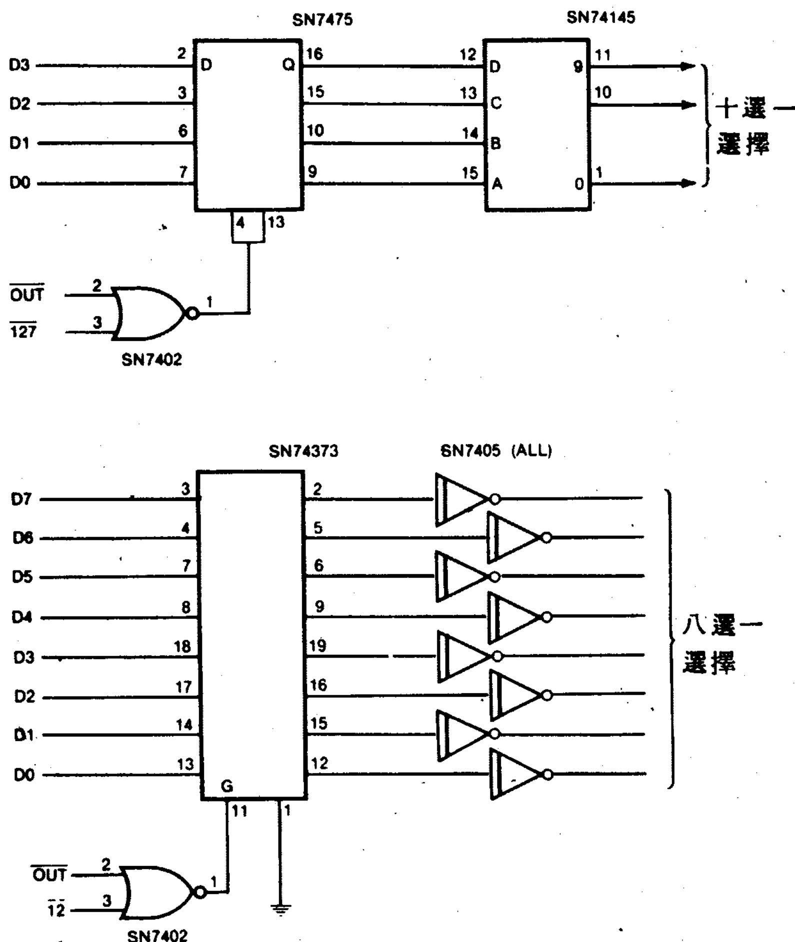


圖 8－10　兩種以鎖住輸出口控制集極開路式元件的方式

接著再接至SN74145解碼器，只要相關的二進值呈現在解碼器的輸入，則解碼器的輸出即可一次一個地動作。使用一個這樣的電路，計算機即可長期地打開或關閉各種設備。不過，記得，只要使用解碼器電路，則每次即僅有一個接至解碼器輸出端的設備能受控制。倘若計算機所控制的某一設備欲獨立，或甚至同時動作，則這個設備就必須使用個別的輸出。這時，只要每一個輸出所用的都是集極開路式電路，您即可以一簡單之鎖住器電路達成此一功能。在這樣的結構內，一個八位元的鎖住器即可控制到八個設備。由於每一位元位置彼此相互獨立，故八個設備可獨立控制。圖8－10所示乃兩種不同型態的ロ控制集極開路式界面。

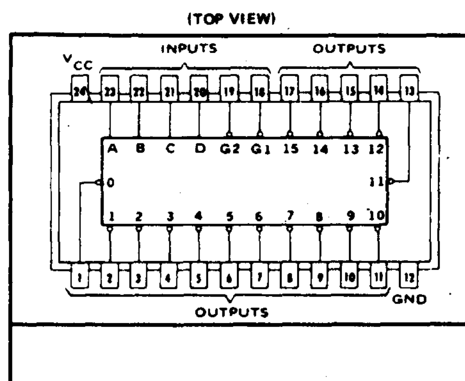


圖8－11　SN74145集極開路式4線對16線解碼器晶片的接腳圖

如圖 8 －11所示，SN74159解碼器乃一 4 線對16線的解碼器。這塊積體電路乃 SN74145 之高電流與高電壓代替品，其於控制如圖 8 － 6 所示之界面電路上非常有用。SN74159 解碼器產生脈衝的方式類似於前面提過的 SN74145（圖 8 － 8 與 8 － 9 ）。

當小型的計算機系統欲擴接額外的記憶器，致好多記憶容量（位元組數）不同的記憶晶片欲混合在一起時，您亦會用到 SN74159。舉個例子而言，記憶晶片有 1 K（ 1024 ），2 K與 4 K位元組等規格。在使用這麼多種記憶晶片的系統裡，我們即可以一 SN74159解碼器輕易地產生這些不同"大小"之記憶晶片所需的所有晶片選取信號，因為，SN 74159 的解碼輸出可連在一起。圖 8 －12所示之解碼方式即以記憶位址線解碼現有的記憶位址，將之全部分成 1 K的區段。然後，由於 2 K 的區段（記憶晶片）需要兩個晶片選信號，4 K的區段需要四個晶片選取信號，因此，這些信號就分別直接連在一起，加至適當的晶片上，而不必再另以邏輯閘電路閘控制。當然，這樣所產生的三個晶片選擇信號，每一個都必須供給一提升電阻，致使在解碼器之輸出不動作時，每一條實際均產生邏輯 1 。SN74159 集極開路式解碼器同樣可用於其它許多電路。只要您隨時記得，標準 SN 7400 系列解碼器的輸出無法接在一起，而唯有集極開路式邏輯閘的輸出可彼此連接在一起，以產生這些閘控功能就夠了。

平常較常用的最後一種集極開路式電路，就是 SN74141 十進解碼器。這個解碼器特別製造成十選一的結構，以推動

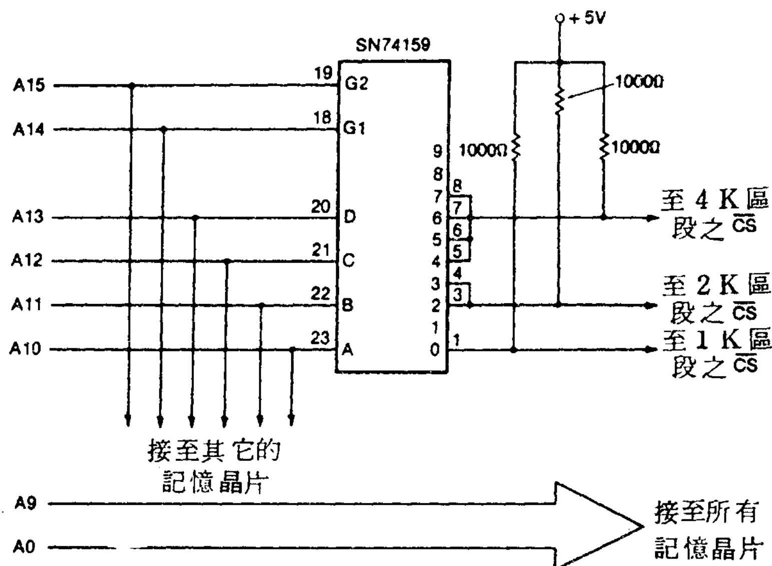


圖 8－12 以一集極開路式解碼器產生數個記憶區段之晶片選取脈衝

高電壓、充氣體、冷陰極之顯示管。SN74141 解碼器典型上均用於推動充氣顯示器之電路上，諸如霓虹燈，或複雜如 Nixie® 管之數字顯示器等。雖然目前較新型的設計已不再採用這種顯示元件了，但在較老的設備上却仍經常看到。SN74141 用以推動個別的充氣顯示器，如 NE－2 與 NE－2H 霓紅燈，或顯示盤等。在作高電壓顯示器界面時尤須特別小心，因爲，一不小心，高電壓就會碰到錯誤的印刷電路板接點，致將之燒毀。由於充氣顯示器導電時幾乎成無電阻狀態，故使用這種顯示器時，一定要記得加上串聯的限流電阻。

SN7400 系列還有許多其它的集極開路式電路，不過，

這些大部份都是用以推動七段ＬＥＤ顯示器的七段解碼推動電路。當然，還有其他的閘控電路，但這些電路通常均不如本章所介紹的這些這麼常用。

8－3　週邊推動器與電晶體陣列

這一節將介紹若干爲使ＴＴＬ吻合元件能用於控制繼電器，ＬＥＤ，與白熾顯示器，所特別設計的ＩＣ，並說明一些可能用法。週邊推動器電路指的是那些將推動電晶體與一邏輯閘合做在單一塊電路上者。每一塊ＩＣ包裝內通常都有兩個週邊推動器，而且電晶體亦可，或不直接連接至邏輯閘輸出。使用NAND，NOR，AND，或OR 閘，推動器可組成各種不同的結構。同樣地，藉著使用設計作特殊用途的電晶體，電晶體陣列亦有多種不同的結構。不過，這節我們將僅討論一般計算機界面上常用的電晶體陣列。

週邊推動器

週邊推動器指的是含有一邏輯閘以及一推動電晶體的IC。視所選用之推動器或推動器所作之應用的型態不同而定，電晶體與邏輯閘之輸出可連接在一起，亦可不連接在一起。這些晶片最典型的代表可能就是德州儀器（ＴＩ）公司的SN 75400族，表１－３所摘列的，即爲此些元件的電流推動

能力與電壓極限。大多數情況下，較小的族都可選擇以 NAND，NOR，AND，或OR 閘控制推動電晶體（或輸出電晶體）。圖 8 —13所示即為每個晶片上具有兩個推動器之週邊推動器晶片的例子。圖 8 —14所示則為 SN75450 B ，SN75460，以及 SN75470 晶片的功能圖與接脚圖。雖然這三種元件的接脚完全相同，但如表8 — 3 所示的，其電流推動範圍則相互迥異。

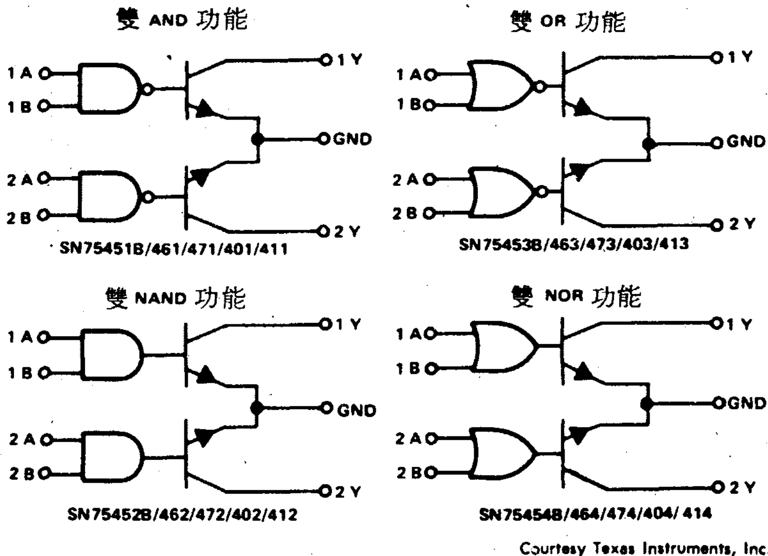


圖 8 —13　一些典型雙付週邊推動器晶片的邏輯電路圖

注意到，輸出電晶體連接至邏輯閘的雙倍週邊推動器極類似於前面所提過的集極開路式元件。週邊推動器所取用的電流高於集極開路式元件，電壓亦然。圖 8 —15所示即為SN 75451 B 推動器之一典型應用。由圖中可看出，推動器晶片

表 8－3　SN75400系列週邊推動器晶片之特性

最大裁止電壓	最小鎖住電壓	最大輸出電流	典型延遲時間	輸出截波二集體	所含推動器數	輸入吻合性	元件型態與包裝		邏輯功能
							−55°C To 125°C	0°C To 70°C	
15 V	15 V	300 mA	15 ns		2	TTL, DTL		SN75430 J,N SN75431 JG,P SN75432 JG,P SN75433 JG,P SN75434 JG,P	AND* AND NAND OR NOR
30 V	20 V	100 mA	22 ns		2	ECL		SN75441 J,N	OR
30 V	20 V	300 mA	21 ns		2	TTL, DTL	SN55450B J SN55451B JG SN55452B JG SN55453B JG SN55454B JG	SN75450B J,N SN75451B JG,P SN75452B JG,P SN75453B JG,P SN75454B JG,P	AND* AND NAND OR NOR
35 V	30 V	300 mA	33 ns		2	TTL, DTL	SN55460 J SN55461 JG SN55462 JG SN55463 JG SN55464 JG	SN75460 J,N SN75461 JG,P SN75462 JG,P SN75463 JG,P SN75464 JG,P	AND* AND NAND OR NOR
35 V	30 V	500 mA	33 ns		2	TTL, DTL		SN75401 NE SN75402 NE SN75403 NE SN75404 NE	AND NAND OR NOR
50 V	50 V	350 mA	1 μs	YES	7	TTL, DTL, CMOS, P-MOS 14-V to 25-V P-MOS TTL and 5-V CMOS 6-V to 15-V P-MOS, CMOS		ULN2001A† J,N ULN2002A† J,N ULN2003A† J,N ULN2004A† J,N	INVERTING BUFFER

70 V	55 V	300 mA	33 ns		2	TTL, DTL	SN55470 J SN55471 JG SN55472 JG SN55473 JG SN55474 JG	SN75470 J,N SN75471 JG,P SN75472 JG,P SN75473 JG,P SN75474 JG,P	AND* AND NAND OR NOR
70 V	55 V	300 mA	100 ns	YES	2	TTL, DTL, MOS		SN75475 JG,P	NAND
70 V	55 V	300 mA	100 ns	YES	2	TTL, DTL, MOS		SN75476 JG,P SN75477 JG,P SN75478 JG,P SN75479 JG,P	AND NAND OR NOR
70 V	55 V	500 mA	33 ns		2	TTL, DTL		SN75411 NE SN75412 NE SN75413 NE SN75414 NE	AND NAND OR NOR
70 V	55 V	500 mA	100 ns	YES	2	TTL, DTL, MOS		SN75416 NE SN75417 NE SN75418 NE SN75419 NE	AND NAND OR NOR
100 V	60 V	350 mA	130 ns	YES	7	TTL, DTL, CMOS, P-MOS 14-V to 25-V P-MOS TTL and 5-V CMOS 6-V to 15-V P-MOS, CMOS		SN75466† J,N SN75467† J,N SN75468† J,N SN75469† J,N	INVERTING BUFFER

*輸出電晶體基極外接至邏輯閘輸出。

†0°C to 85°C

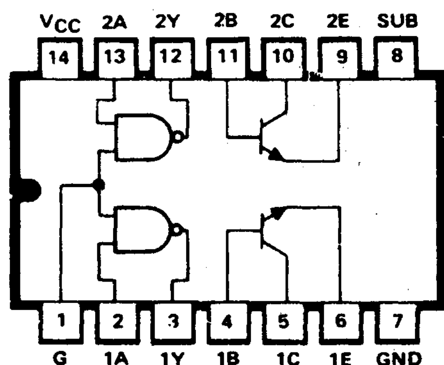


圖 8－14 SN75450B,SN75460與SN75470週邊推動器晶片之功能圖與接腳圖

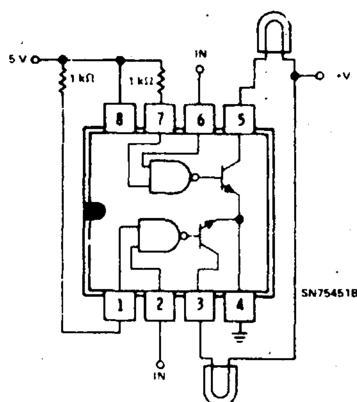


圖 8－15 以SN75451B推動器晶片推動兩個燈泡

分別推動了兩個白熾燈泡。圖 8－16所示則爲另一種應用。這種例子以一SN75475推動器控制兩個48伏特的繼電器。由於這個晶片本身即含有截波（或稱傍路）二極體，因此，用者不需再外加保護推動電晶體的二極體。

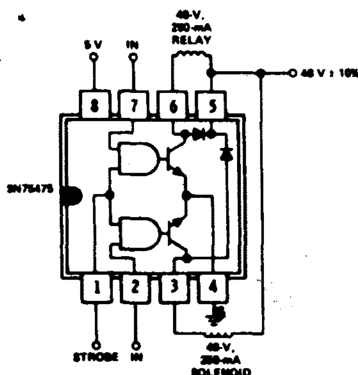


圖 8 －16　以SN75475週邊推動器推動兩個繼電器

有許多場合亦不適合採用ＬＥＤ作指示燈，譬如，若觀察者欲遠遠觀看，或必須能看出高準位與低準位的發光狀態，由各種角度觀看，或甚至需要各種色彩等。由於ＬＥＤ無法完全滿足這些需求，因此，這種情況下通常還是採用白熾燈泡。白熾燈泡所需之電流比ＬＥＤ高，而且推動電壓經常亦是，一般均在12至28伏特之間。由於白熾燈泡在燈絲冷却時有入侵電流，且這個電流通常高達一般動作電流的十倍以上。因此，當以週邊推動器晶片推動白熾燈泡時必須特別小心。目前已有許多方法可以減低入侵電流，以下我們即介紹其中幾種。降低入侵電流最簡單的方法就是將燈絲保溫。這只要在燈泡與地之間，用一個“保持動作”的電阻器與推動電路並聯，即可輕易達成。圖 8 －17所示即爲這樣的情形。其中，電阻值必須選定在使流經燈泡的電流爲燈泡平常之全動作電流的十分之一處。圖 8 －17的電路即選用了 6 伏特，150 mA 的燈泡，以及 150 歐姆的電阻。這個電阻值正好

可使燈泡流過約40 mA 的電流，但却不發亮。保溫電阻的功率損耗額定一般均爲½或 1 瓦特。

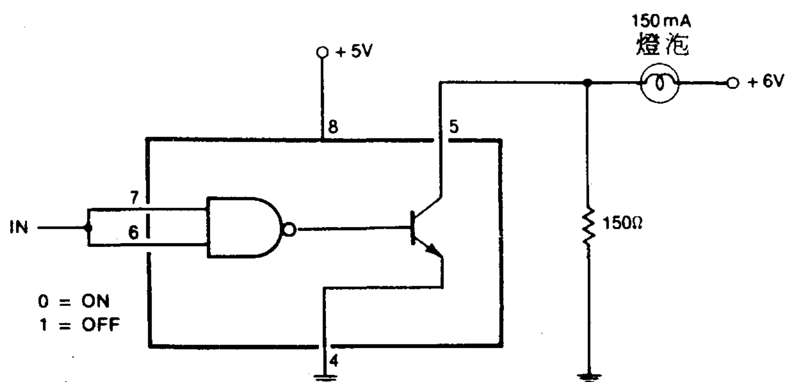


圖 8 —17　以SN75451A作白熾燈泡推動器，附加一保溫電阻

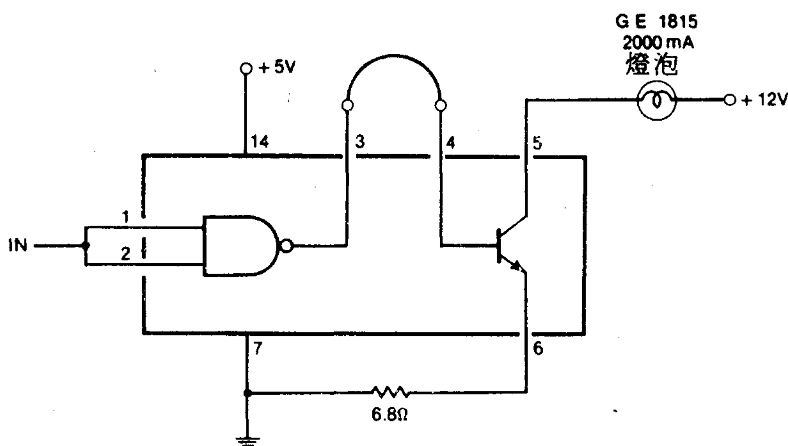


圖 8 —18　使用SN75450B推動器及限流電阻

爲了限制燈泡一亮時之峯值電流，我們亦可以一限流電阻與燈泡串聯，這種情形即如圖8－18所示。這個電路以SN75450 B作推動器，且邏輯閘之輸出直接接至電晶體之基極輸入。您應記得，在這種電路內，眞正限流的是電晶體與限流電阻兩者，而非單只限流電阻。限制電流並不難求，其主要等於射極（emitter）電壓除以射極電阻。射極電壓又等於邏輯閘之輸出電壓（3.3伏特）減電晶體之射一基偏壓（0.95伏特）：

$$I_L = \frac{V_{OH} - V_{BE}}{R_E}$$

$$= \frac{2.35\text{ 伏特}}{6.8\text{ 歐姆}}$$

$$= 0.345\text{ 安培或 }345\text{ 毫安}$$

電晶體在這一點達到飽和，將電流限制於約340 mA。由於燈絲開始很快地加熱，因此，電流亦開始下降至正常的大小，就電路圖中所示之GE－1851型燈泡而言，約爲200mA。最大波動電流不可超過SN75450 B推動器之最大波動準位，同時，平常在導電狀態時，這個電流亦不可受電晶體飽和的限制。因此，公式所定義之波動電流限制，實際應高於燈泡亮時所流經燈泡之正常電流。

另一種使白熾燈泡發亮的辦法，是使用兩個推動器電路，一個作正常轉換，另一個作電流限制轉換。圖8－19所示

之例子以一 390 歐姆之電阻以及一 500 μ F之電容組成一單穩定電路。當燈泡之輸入控制線變爲邏輯 1 時，限流電路即開始經燈泡導電。在 RC 網路"計時"時間到後，限流網路即爲另一個推動器直接接地所傍路。於圖 8 －19之電路，在第二個推動器直接接地前，燈泡約有 200 ms 的"熱溫"時間。

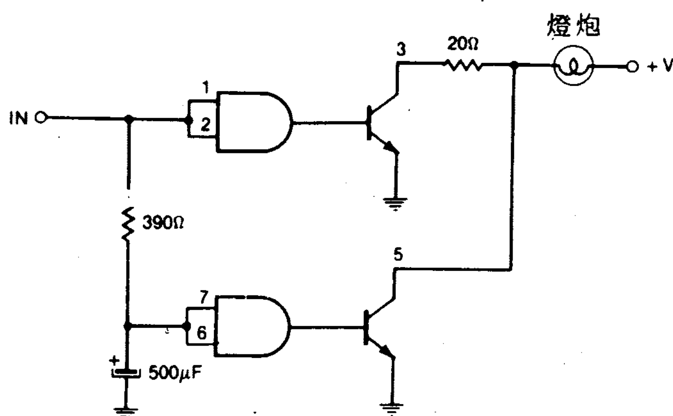


圖 8 －19　以兩個SN75452A推動器作燈泡推動電路的電流控制

繼電器的推動比燈泡推動稍稍簡單一點。誠如前面所說的，繼電器推動電路通常以一二極體與繼電器之線圈並聯，二極體之極性正好與電流反向。這個二極體的功用即在傍路繼電器關閉瞬間所產生的反電動勢。同樣地，輸出電晶體亦須作適當的保護。電晶體在截止時，集極電壓幾乎瞬間升高至推動繼電器線圈的電壓。（這個電路的電晶體集極接至線

圈，射極接地，基極直接接邏輯閘輸出。）若集極迅速地升至這個電壓，則推動電晶體可能會燒毀。因此，為了減緩集極電壓昇高的速度，在電晶體不導電後，我們在電路之集極與地之間加上一很小的電容。圖 8 — 20所示即為加上這種電容的一個典型電路。若週邊推動電路已內含有保護二極體，則圖中所示之保護二極體即可免了。

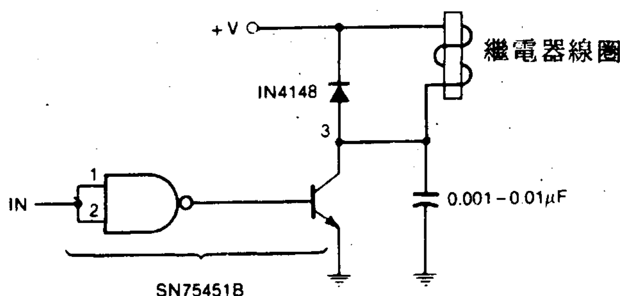


圖 8 －20　完全保護之繼電路推動電器

前面我們曾經說過，有些週邊推動器晶片內之輸出電晶體並不接至邏輯閘上。在這些推動器上，譬如，SN75450 B 就是一個例子，晶片所生產之矽材料 —— 稱為基底（ substrate ）—— 並不接地。因此，只要將ＩＣ之基底接腳接至更負的電壓，電晶體即可用於負的電壓。未接的基極同時亦很好用，因為，我們可在邏輯閘輸出與基極輸入之間接上一個電阻，限制流經電阻的電流。由於作者認為這些應用在正常之界面工作上並不非常重要，所以，一直未舉任何特定的例題。

在討論週邊推動器電路時，還有其它一些重點必須加以注意，由於繼電器、燈泡、LED、與其它元件平常所加之電

源電壓均超過TTL 電路所用的+5伏特，因此，您一定要特別記得，這些電壓絕對不可在TTL電源電壓未加上前先加上。倘若萬一TTL＋5伏特電源不加而僅加上這些較高的推動電壓，則必須有某種方法使週邊晶片享有自己的電源電壓。其中最簡單的方式或許就是由這些用以推動繼電器與燈泡等之較高電壓，獲得週邊推動器晶片所需之邏輯供應電壓。這時，我們可使用一簡單之電壓調節器。如此，只要系統一加上推動電壓，週邊推動器即有邏輯供應電壓。個別的TTL晶片可使用個自的＋5伏特電源。

有關週邊推動器電路的進一步資料，請您參考德州儀器公司之" The Peripheral Driver Data Book"（週邊推動器資料手册）以及" Linear and Interface Circuats Appl-ications "（線性及界面電路應用）兩本書。

電晶體陣列

有許多種不同的電晶體陣列可立即應用至必須控制高電流與高電壓之界面上。電晶體陣列之實際例子包括RCA公司之CA 3096AE ，這是一塊含有三個npn以及三個pnp電晶體的IC，以及將在後面詳述，諸如Sprague ULN-2800 A系列晶片等之更複雜元件。這些電晶體陣列的主要特性，是其可輕而易舉地與TTL 吻合電路相界面，以產生推動繼電器、LED、七段顯示器、白熾燈泡，以及其它週邊設備之能力。其中有些電晶體陣列必須加上一點額外的零件，以產生適當的輸

入電壓與電流準位等等。在您研究您的設計是否可能採用電晶體陣列時，我們建議您仔細地檢討一下您的界面要求。有許多應用確實是採用電晶體推動器比採用週邊推動器晶片或集極開路式晶片更適合。絕大多數電晶體陣列均無像週邊推動器晶片所附的NAND，OR，NOR或AND 邏輯閘。因此，若同樣晶片上不需用到這些閘控功能，則電晶體陣列即應被仔細考慮。事實上，一已知ＩＣ封裝內所含的經常還勝於相當之週邊推動器電路呢！

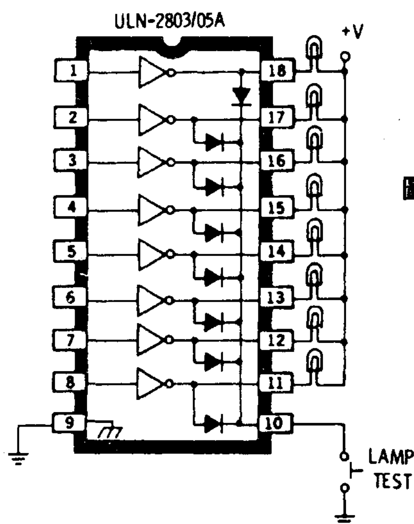


圖 8 －21 以ULN-2803/05A作燈泡推動器的電路圖

圖 8 －21所舉之例題即說明了以— Sprague ULN-2803或-2805 推動器推動八個白熾燈泡的用法。由於晶片本身已含有保護二極體，故這個電路巧妙地將之用在燈泡測試特色上。只要燈泡測試開關一關，電流就會流過八個二集體，使每一個燈泡發亮。若燈泡不亮，即表示燈絲已燒掉。表8-4

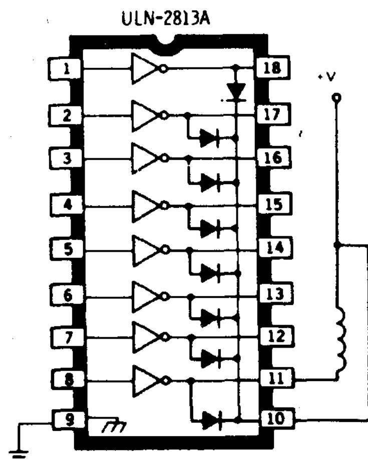
列出了ULN-2800A族內的許多推動器。由表可看出，這個族內含有可用於TTL，CMOS，PMOS 等各種邏輯族與不同工作電壓的各種不同元件。圖8－22所示即將ULN-2813A用作一繼電器推動器的用法。注意，晶片所內含之保護二極體已接至＋V電源供給，以傍路繼電器關閉時所產生之反電動勢。

表 8－4　Sprague ULN-2800族之特性

$V_{CE(MAX)}$ = $I_{C(MAX)}$ =	50 V 500 mA	50 V 600 mA	95 V 500 mA
	型　號		
General Purpose PMOS, CMOS	ULN-2801A	ULN-2811A	ULN-2821A
14-25 V PMOS	ULN-2802A	ULN-2812A	ULN-2822A
5 V TTL, CMOS	ULN-2803A	ULN-2813A	ULN-2823A
6-15 V CMOS, PMOS	ULN-2804A	ULN-2814A	ULN2824A
High Output TTL	ULN-2805A	ULN-2815A	ULN-2825A

圖8－23所示是一讓一堆LED其中之一LED發亮的應用。電路以一SN74145解碼器以及集極開路式輸出，產生每一列LED所取用之電流，並以一RCA CA3082產生每一行LED所需之電壓，只要供應SN74145 解碼器一個列選擇碼，以及提供CA3082 一個行推動信號，這個電路即會選取一個LED注意到，行選擇並未解碼。雖然表面看來這像是一個簡單的例題，其中之二極體陣列可能相當於一通用儀器公司所生產之MAN-2A或相當於文數字顯示所用的5×7LED

圖 8－22　以ULN-2813A作繼電器推動器。注意到內部保護二極體。



Courtesy Sprague Electric Co.

圖 8－23　以CA3082推動器及SN74145解碼器控制－LED顯示矩陣

陣列。當然，若計算機欲控制的是一LED 顯示矩陣，則必須要有一鎖住輸出口。七段顯示器的控制方式通常亦類似，電流取用使每一節段亮或暗，而電流供給一個接一個地選取每一個顯示數字。

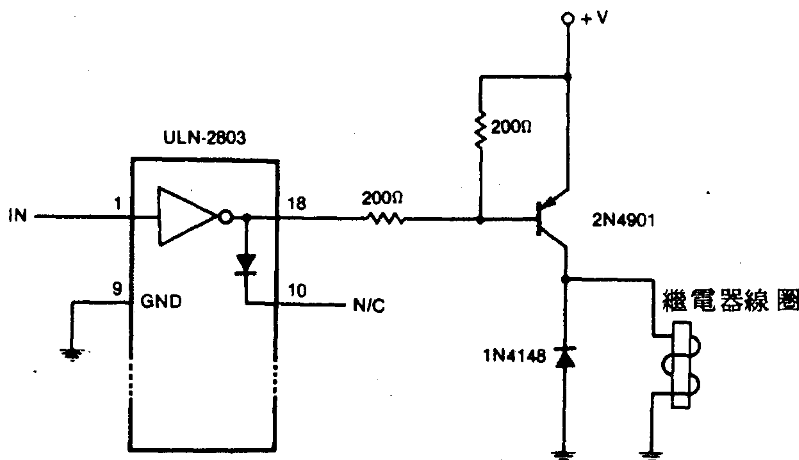


圖 8 —24 以一外部電晶體放大電晶體陣列之電流推動能力

有時，界面電路所需求之電流可能超出電晶體陣列晶片上之電晶體的電流額定。在這種狀況下，我們就必須另外加一外部電晶體，將界面電路之電流推動能力進一步放大。圖 8 —24所示，即以一 2 N 4901 電晶體推動 ULN - 2803陣列之其中之一電晶體的情形。 2N 4901 之最大集極電流可達 5 安培。

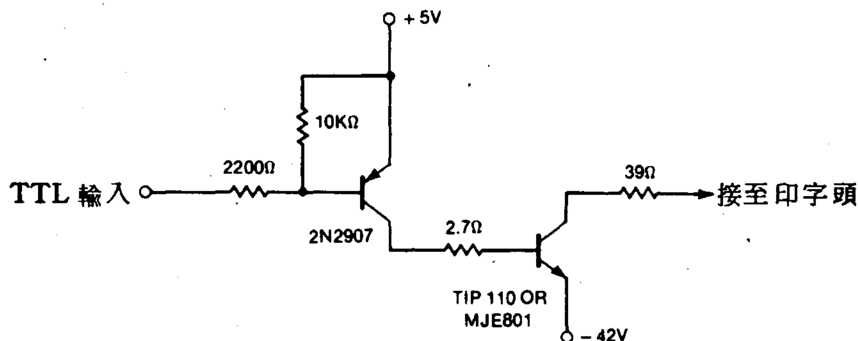
若干靜電式的印字機使用鋁化紙。當印字頭送一高電壓脈衝至鋁化之紙表面時，紙表面之鋁即蒸發，留下一可見之

黑點。這些點以 5 × 7 短陣之格式組合，形成字母、數字、或特殊符號。雖然您不必管這些點之各行各列在界面內如何產生，但您却必須設計在一接獲命令後，能送出一負42伏特脈衝給印字頭的電路。這似乎又可使用電晶體陣列或週邊推動器晶片。事實上，有好幾種電路可解這個問題。圖 8 — 25所示即爲控制電極，以產生鋁紙所需之 — 42伏特脈衝的各種方式。圖中皆僅劃出一組電路，事實上，總共必須有七個同樣的電路，才能推動七列之點陣印字機。TI 之 TIP110 與Motorola 之 MJE 801電晶體均爲達靈頓（ Darlington ）對。ULN - 2003 A 爲 Sprague 50 伏特， 500mA 之推動器，而 DS 8897 則爲國際半導體公司之氣體顯示器推動電路。對於這種特定的界面問題，將來可能會發展出更簡單的解。

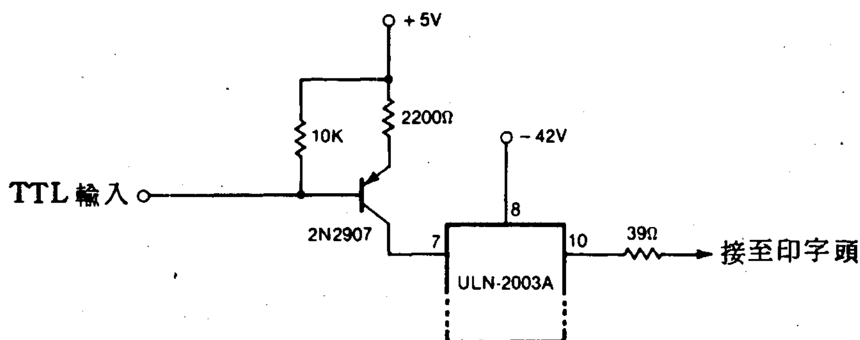
特別記住，前面所介紹過之週邊推動器以及電晶體陣列全部都有電位降。因此，在導電通過高電流時，這些元件一概都會發熱。有些 I C 上就具有散熱用的接脚，可代換上幾個未用的電接脚。若有這些設備設施，那一定要裝好，以便空氣能輕易流通，產生冷却。否則，萬一元件發熱過度，則其載流能力勢必會降低，因而無法達到規格說明書中所列的電流額定。若您決定採用這些元件，千萬記得一定要有有效的冷却。否則，元件將很容易燒毀。

8 — 4　可選取之推動器

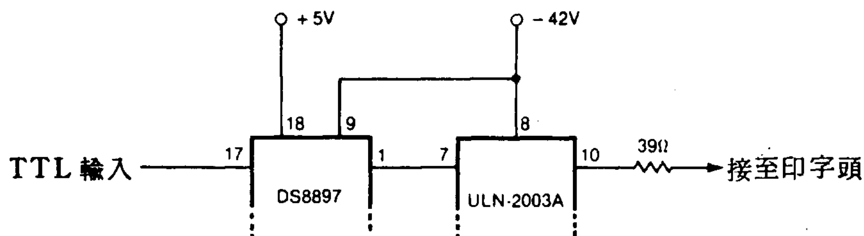
有一系列很好用的界面電路您可能會用到。由於晶片內



(A)TIP110 或 MJE801 射極接－42 V。



(B)ULN－2003 A 之第 8 脚接－42 V。



(C)ULN－2003 A 之第 8 脚接－42 V。

圖 8－25　三種靜電式印字機之印字頭推動電路，顯示了高電壓推動問題之各種解法

均具有鎖住器、解碼器、以及高功率之電晶體輸出，因此，這些元件均稱爲可選取之鎖住器或可選取推動器。在Signe-tics 所製造的元件內，每一個鎖住器鎖出均能流入或流出約300mA 的電流。現有的兩個元件是，電流取用晶片NE590，以及電源產生晶片NE591。這兩個可選取鎖住器晶片均具有八個獨立控制的輸出。由於八個位元中之每一位元均可單獨被選取，且被清除爲0（邏輯0）或設定爲1（邏輯

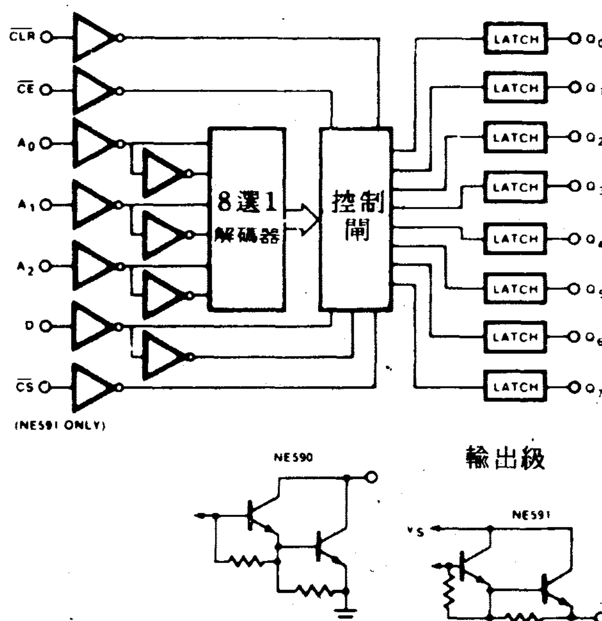


圖8－26　NE590/591可選取鎖住/推動器晶片之方塊圖

1），是以，鎖住器乃可以選取。圖8－26所示即爲NE590／NE591之適用方塊圖，而圖8－27所示則爲兩個晶片之接脚圖。由圖8－27可看出，NE590 爲16支接脚包裝，

而NE 591 爲 18 支接脚包裝。撇開電流之取用與源出不談，這兩個晶片的主要區別是NE 591 另以兩支接脚分別作晶片致能電路之控制，以及連接至正電流源供給。大部份情況下，電源供給電壓均爲十5 伏特，與邏輯電路所用者相同，但有時，其亦可高至十7 伏特。

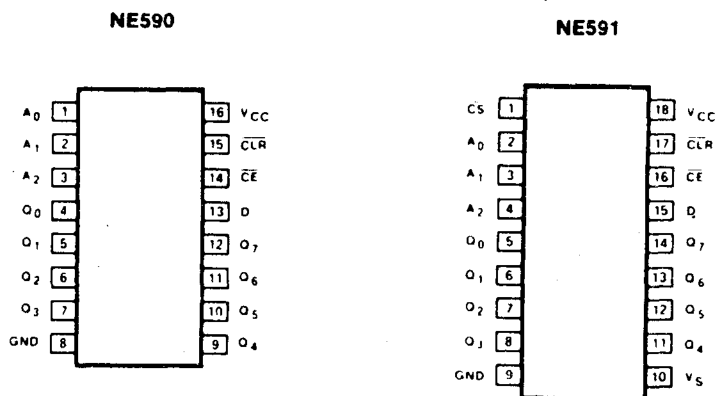


圖 8 －27 Signetics NE590與NE591 可選取鎖住/推動器晶片之接脚圖

除了不是將八位元字組並行地輸出，而是每次以三個位址輸入選擇一個個別位元，並鎖住被選定的位元（邏輯 1 或邏輯 0 ）外，NE 590／NE 591的動作宛如一正常的鎖住器電路。依此，八位元之每一位元均可個別被設定爲 1 （打開）或清除爲 0 （關閉），而絲毫不受其它位元之影響。由於鎖住器電路僅有三個位址位元輸入，因此，勢必還須另有一晶片致能（ $\overline{CE}$ ）輸入。這個輸入可以其它之位址位元與一諸

如$\overline{\text{OUT}}$ 之功能脈衝閘控而得。NE 590 與 NE 591 均單獨設有一清除（ CLR ）輸入，使得八個輸出於控制系列開始之初，或當界面電源一打開之際，能個別被清除爲 0 。

圖 8 — 28 所示即爲一以 NE590所構成的典型界面電路。這個電路將 D_0 資料巴士線上的邏輯準位，鎖入由 $A_2 \sim A_0$ 等位址位元所選定的特定位元內。注意到，其餘的低半位址位元 $A_7 \sim A_3$ 與 OUT （ Apple Ⅱ上相當於 POKE ）功能脈衝經閘控產生控制實際鎖住過程之晶片致能衝脈。這個電路總共需要 8 個不同的設備位址，每一位元一個，而每一輸出命令所傳輸之資料字組的最低次位元決定每一被選取之電

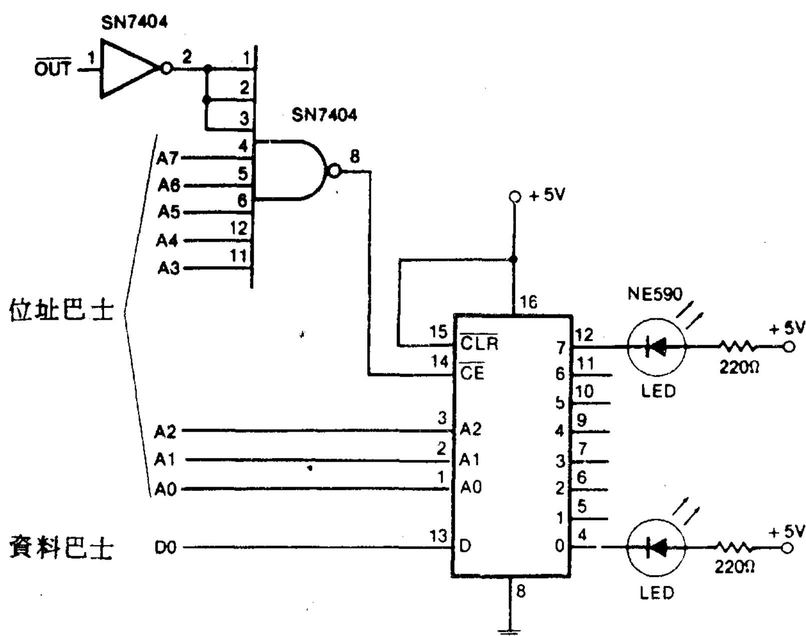


圖 8 —28　使用NE590(電流取用)之典型 LED推動界面

圖 8－5　控制NE590(見圖 8－28)之輸出指令

電晶體	開指令	關指令
Q0	OUT 248,1	OUT 248,0
Q1	OUT 249,1	OUT 249,0
Q2	OUT 250,1	OUT 250,0
Q3	OUT 251,1	OUT 251,0
Q4	OUT 252,1	OUT 252,0
Q5	OUT 253,1	OUT 253,0
Q6	OUT 254,1	OUT 254,0
Q7	OUT 255,1	OUT 255,0

附註：OUT指令在Apple II 上相當於 POKE 指令

表 8－6　控制NE591(見圖 8－29)之輸出指令

電晶體	關指令	開指令
Q0	OUT 96,0	OUT 104,0
Q1	OUT 97,0	OUT 105,0
Q2	OUT 98,0	OUT 106,0
Q3	OUT 99,0	OUT 107,0
Q4	OUT 100,0	OUT 108,0
Q5	OUT 101,0	OUT 109,0
Q6	OUT 102,0	OUT 110,0
Q7	OUT 103,0	OUT 111,0

晶體開關應導通或截止。（於NE 590／NE 591，邏輯 1 時電晶體導通，邏輯 0 時電晶體截止。）就圖 8－28之例子而言，表 8－5 所示之指令即可用以控制可選取鎖住器。不論那一種情況，您隨時都要記得提供正確的資料字組，以使相對的電晶體能如願地導通或截止。傳統鎖住就是一次八個資料位元同時取入，而可選取鎖住器則每次僅取入一個位元，記

住這個區別。

圖 8 －29所示則爲另一種以NE 591供應顯示器所需之電流的界面電路。這個電路之鎖住器資料輸入並不是接至資料巴士而是直接接至其中之一位址位元 A_3。如此，鎖住器所鎖住的位元是邏輯 0 或邏輯 1 即由位址位元 A_3 決定，相對之輸出電晶體亦由 A_3 控制。這時，輸出指令亦不必提供一個特定的資料位元。表 8 － 6 所示即爲圖 8 －29電路的一系列控制指令。您可看出，這個表格上的所有資料值均爲 0。由於資料對鎖住器不影響，因此，這時候資料值是多少均無妨。

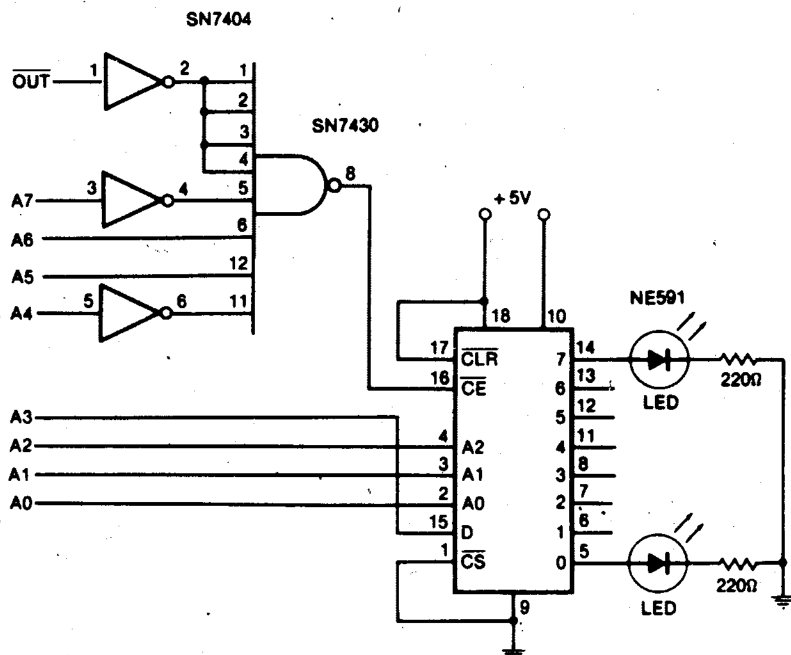


圖 8 －29　採用NE591(供應電流)之LED推動界面

NE590／NE591晶片的輸出應被視爲簡單的電流耗與電流源。這些輸出可以用推動燈泡，LED、繼電器、顯示器，以及其它需要高電流的設備。由於鎖住器的輸出可個別控制，故其提供一種產生控制脈衝之簡易方式，這種控制脈衝可用以控制諸如圖8－5所示之儀器等外部設備。我們曾如此以NE590控制過Diablo Hytype Ⅰ型印字機的一部界面電路。正常的TTL元件並無這種電流推動能力，同時，其亦無法在一個晶片內提供全部所需的功能。

8－5 控制交流線負載

您的計算機系統可能亦有機會控制取用交流117伏特，或平常所謂的線電壓（line voltage），之負載。然而，前面所介紹過之緩衝器、集極開路式晶片、以及電晶體陣列並不能控制高電壓、高電流之負載。這時候，一種最簡單的辦法是教計算機控制緩衝器晶片，然後，緩衝器晶片再控制繼電器，而由繼電器進一步控制需求線電壓之負載的功率。這當然是一種解法，而且事實上若干系統亦採用了這種辦法，

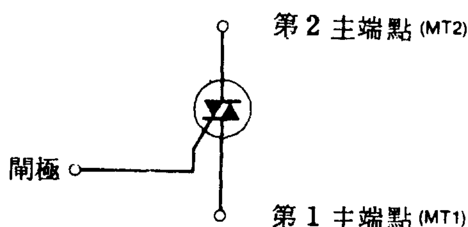


圖8－30 雙向矽控開關元件的表示符號

但還有其它更簡單且更可靠的辦法可達成相同的目的。這一節，我們將探討一些使小型計算機能控制 117 伏特之交流負載的電路。

簡單的固態控制器

或許，可用以控制 117 伏特之交流負載的最簡單電子開關，就是如圖 8－30所示的雙向矽控開關（ TRIAC ）了。當然，半導體開關還有許多其它型式，不過，此地我們的討論將僅限於TRIAC 。TRIAC 元件有一控制流經元件之電流的閘極輸入。此一元件的最簡單用法即如圖 8－31所示。在這個電路內，只要TRIAC 閘極之開關打開（ open），TRIAC 就永遠不會導通。但只要這個開關合上，TRIAC 即立即導通。在界面時，這個開關最好由計算機控制，以期能控制流經負載的電流。圖 8－32所示即爲這種計算機控制之線開關的一個例子。圖中之電路使用三個 TRIAC 元件。

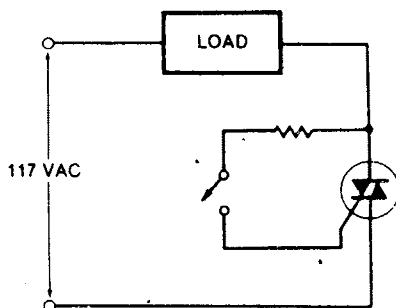


圖 8－31　簡單以TRIAC組成之電機開關的電路圖

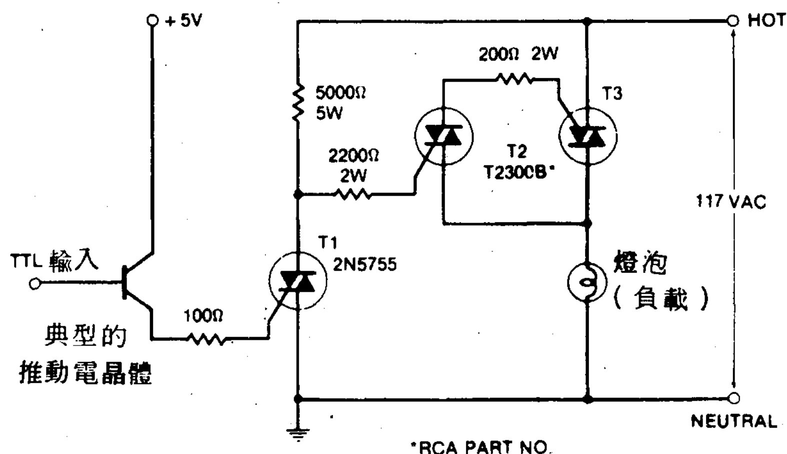


圖 8－32 邏輯控制之 TRIAC開關的電路圖

其中，您應可看出，主TRIAC 爲 T_3 ，其用以控制流經燈泡（負載）的電流。而第二個TRIAC（ T_2 ）的功用即在取代圖 8－31中之開關。這個開關用TRIAC 又受另一個T TRIAC（ T_1 ）的控制。最後，最左邊之邏輯輸入使電晶體導通或截止，控制了以TRIAC 構成的開關。電路內所使用的＋5 伏特電源爲邏輯供給電壓。由於以白熾燈泡作負載，因此，電路增加了其它的零件控制入侵電流。

舉這個簡單例子的目的乃在說明如何以TRIAC 元件控制線負載。由於有更簡單的電路可達成同樣的開關功能，因此，這個電路的動作原理並不需作更詳細的解說。圖 8－32的電路可能有一點讓您頗覺困惑。那就是，計算機與 117 伏特之交流線電壓之間有直接的電連接。圖 8－32所示的電路平常應可正常動作，但萬一有人不小心將螺絲起子或其它導

電的物體掉落在電路內，則計算機的電路與線電壓之間很可能就會構成連接，進而損壞計算機。因此，我們必須想辦法避免計算機受到這種傷害。

計算機電路與線電壓控制電路間可以—**光耦合器**（Optical coupler ）加以隔離。這種元件的切面圖即如圖 8 —33所示。一紅外線感應的光測器上置有一紅外線的發光二極體（ IRED ）。當 IRED 導電時，紅外線光即通過〝燈管〞到達光測器。入射的紅外線光使電流流經光測器。當然，當 IRED 不得導電時，電流即停止流經光測器，或至少其流動大大地減低。如此，由於 IRED與光測器間並無電連接，因此，使 IRED 導通與不導通的輸入信號與光測器彼此隔離

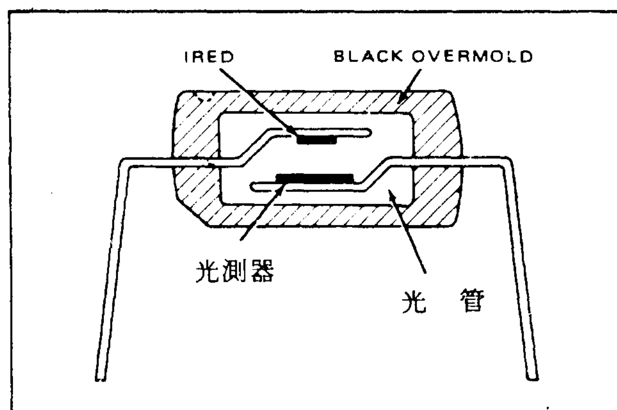


圖 8 —33　光交連晶片之切面圖

Motorola　半導體公司的設計小組已設計出一種能直接與TRIAC 與邏輯準位動作的光耦合器。這個MOC3011元件

使計算機得以控制 IRED ，而光測器所測知之紅外線光再用以控制TRIAC 。據此，不必作直接的電連接、計算機即可控制到TRIAC電路。圖8 —34所示即爲MOC 3011 之接脚圖。圖8 —35所示則爲兩個簡單的控制電路，這兩個電路舉例說明了TRIAC電路如何以MOC3011 控制電阻性（燈泡）與反動性（馬達）的負載。

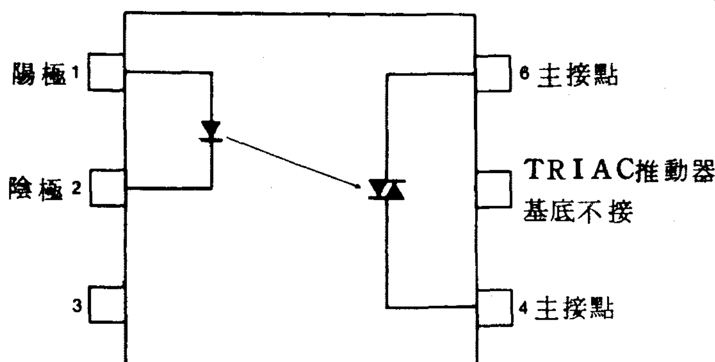
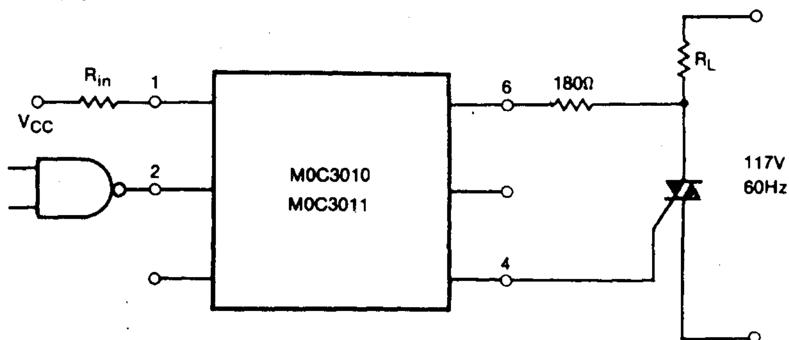
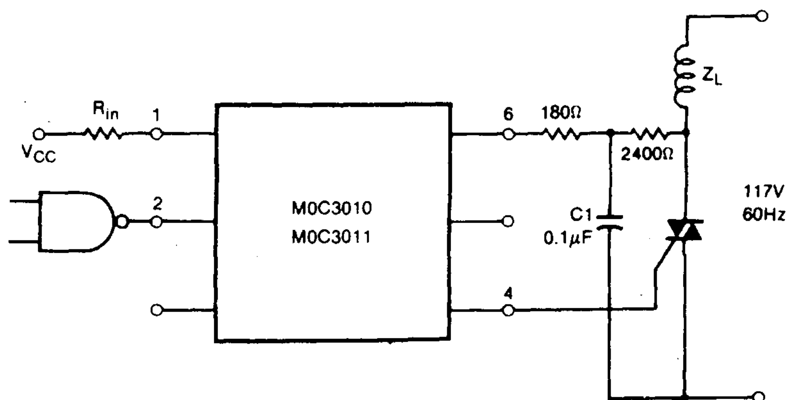


圖8 —34 MOC3011光耦合器之接脚圖

若您打算在許多不同的界面上使用MOC 3011 型的電路，那最好能在光耦合器之 IRED 邊加上保護。這只要加上一簡單的網路便可：一防止反極性連接之二極體以及一限制流經 IRED 之電流的電晶體。這個保護電路正如圖8 —36所示，圖中所示正是一通用的線控制電路。其中，2N6071 B TRIAC可承受AC117 伏特，4 安培之電流。特別記住，TRIAC 兩端之電壓降亦會產生熱。是以，若您選用這個電



(A) 電阻性負載



(B) 反動性負載

圖 8－35　電阻性負載及反動性負載之 TRIAC 控制電路

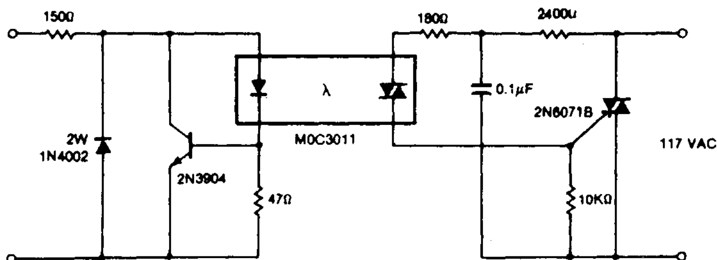


圖 8－36　具有保護邏輯輸入之通用 TRIAC 控制電路

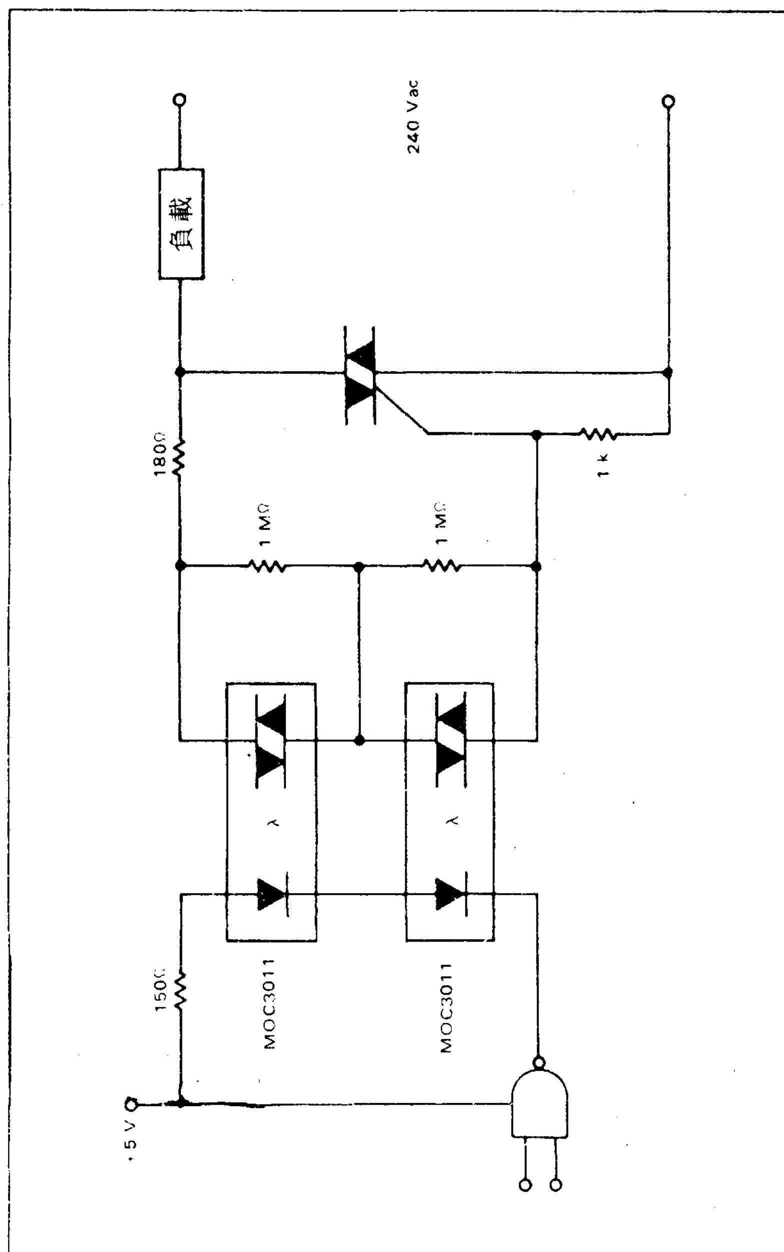


圖 8－37 以MOC 3011 光耦合器控制AC 240V之TRIAC開

路，則2N6071 B最好加上散熱片。不過，萬一您要控制的設備需求更高的電壓，譬如說，240伏特AC，那您就不能單獨使用MOC3011了，因爲，其無法耐這麼高的電壓。但若您如圖8－37所示的，使用兩個MOC3011，使電壓（240V AC）能等分降在每一個上，那就可以了。這個電路以兩個1M歐姆的電阻作分壓電路。爲了清楚起見，輸入端之保護網路在這個圖就不畫出。

固態繼電器

若您不急着構成您所欲控制之每一線負載的TRIAC線開關，則事先包裝好的固態繼電器亦是一極佳的考慮對象。這種封裝內含有一切必要的TRIAC與其它固態電路，您只需將之裝在電路內，並與計算機界面作一些很簡單的連接即可。這些固態繼電器的電路圖極類似於圖8－36所示者，圖8－38所示即爲一些實物的照片。這些元件有各種大小、形狀、以及工作電壓與電流。固態繼電器有優點亦有缺點。表8－7所列即爲其中一些優缺點。絕大部份固態繼電器若不是AC117伏特，即AC220伏特，且繼電器之載流能力最高爲45安培。由於絕大多數固態繼電器與控制負載的TRIAC之間都是光耦合的。故繼電器之控制與線負載兩側均有極佳之電壓隔離。這個電壓隔離通常均在1500至2500伏特之間。

固態繼電器用在界面應用上時必須特別小心。在以這種

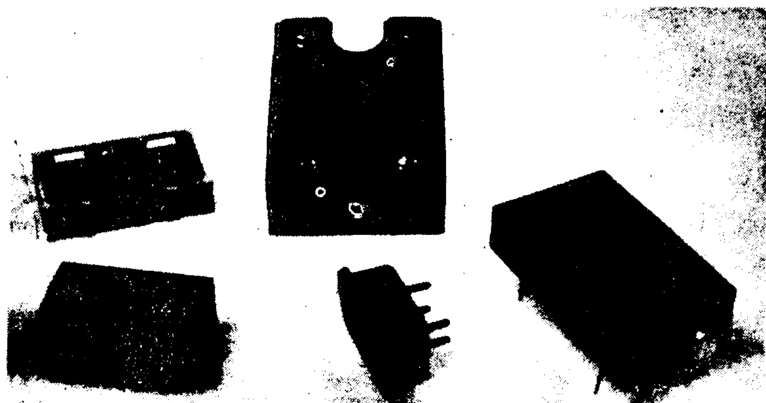


圖 8 —38　各種固態繼電器之照片

表 8 — 7　固態繼電器之優缺點

優　　點	缺　　點
堅固，不會磨損	由於開關半導體有電壓降，故必須散熱。
不需接點清洗	一般只能轉換AC 負載。
直接與 TTL 及其它邏輯族吻合	不易獲得不同的接觸方式
性能可重現	可能很貴

元件控制高電壓與高電流線負載時，我們永遠無法得知負載將在AC117伏特，60HZ（或50HZ）週期的何處導通。因此，若繼電器正好在峯値電壓時導通，則固態繼電器將產生一瞬間之入侵電流。這個高電流經常會超過我們所使用之繼電器的最大電流額定。同時，此一瞬間大電流亦可能使固態繼電器電路放射出大量、頻率廣泛的靜電雜音。計算機電路在汲取這個雜音後，很可能會發生故障，或產生其它毛病。所幸，這個問題可藉著使用零電壓轉換（Zero-uoltage Suit-ching）的技巧，將之大大地削減。使用零電壓轉換時，縱然繼電器之控制邊已在一電壓週期之中間某一點開始動作，但繼電器還是會等到加至負載邊的交流電壓抵達零伏特時，才會開始導電。圖8－39所示即爲交流117伏特之零電壓點，而圖8－40所示則爲導通命令與負載之實際導通之間的時序關係圖。由圖可看出，導通命令至繼電器實際導通間之最大延遲約爲8.3毫秒，相當於60HZ頻率之半週期。絕大多數使用固態繼電器的應用都會有一個無意義的週期。由於採用零電壓轉換的固態繼電器必須增加額外的電路，以測知零

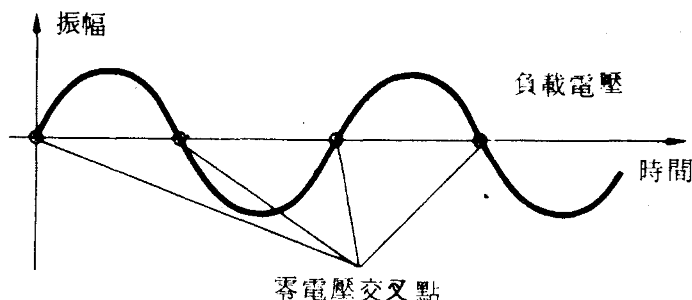


圖8－39　AC 117伏特波形之零電壓點

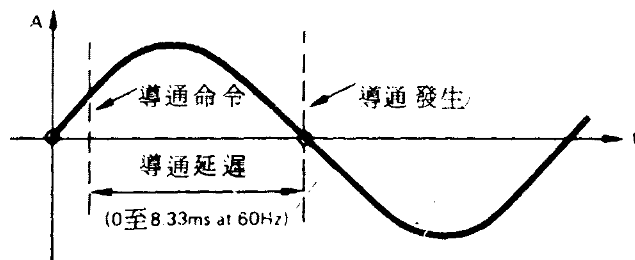


圖 8－40　導通命令與實際零電壓轉換點間之時序關係

轉換點，因此，即使不導電，負載上仍會通過一點電流。但這個電流一般均不超過幾毫安。實際上，絕大多數的零轉換固態繼電器都不是正好在零伏特時轉換，而是在零交叉點之 1 至 5 伏特之間導通。

舉個例子而言，假設有一個控制系統欲以一個固態繼電器啓開與關閉一白熾燈泡。入侵電流的問題在前面雖然已經討論過了，但仍是選擇固態繼電器的重要考慮事項之一。我們假設燈絲的冷電阻爲 2.4 歐姆，且導通電阻爲24歐姆。對白熾燈泡而言，這個比例（ 10： 1 ）相當合理。爲求最壞情況，我們假設燈泡於AC117 伏特之峯值時導通。因此，流經燈泡的瞬間電流約爲70安培：

$$I_{PEAK} = \frac{V_{PEAK}}{R_{COLD}}$$

$$= \frac{120\sqrt{2}}{2.4}$$

$$= 70 \text{ 安培}$$

倘若我們假設繼電器在5伏特左右轉換，則此一瞬間電流將大大減少至2安培左右，而非70安培。由於絕大多數的固態繼電器均能在高電流下動作幾個週期，因此，只要不要維持太久，固態繼電器通常還是能忍受極高的起動與入侵電流的。舉個例子而言，一最大額定電流爲10安培的固態繼電器，通常能容忍50至60安培的電流一至兩個週期，30安培的電流10個週期，或15安培的電流100個週期。

由於閘極信號除去後，TRIAC元件仍會繼續導通，因此，您可能會想，這些TRIAC如何關閉呢？事實上，這個TRIAC自己會處理好好的，因爲，TRIAC的特性就是一直導通至其兩端的電壓降至零爲止。因此，根本不需有任何特殊的關閉電路。

8－6　繼電器保護



固態繼電器由於經常用於如馬達等電感性負載或大型繼電器上，故在關閉時，這些負載可能會回生一個反電動勢。而固態繼電器內的TRIAC可能會因感應到這個反電動勢或電壓脈衝，致其又重新導通。因此，假若有這種問題存在，我們建議您最好在固態繼電器之負載輸出兩端加上一"損減"（snubber）電路。這個損減電路可幫忙降低電壓脈衝，使繼電器不會再重新導通。典型之損減電路即如圖8－41所示。同樣地，電源線上的短暫高電壓脈衝亦會致使繼電器導通

固態繼電器由於使用TRIAC，故其將在負載電路兩端

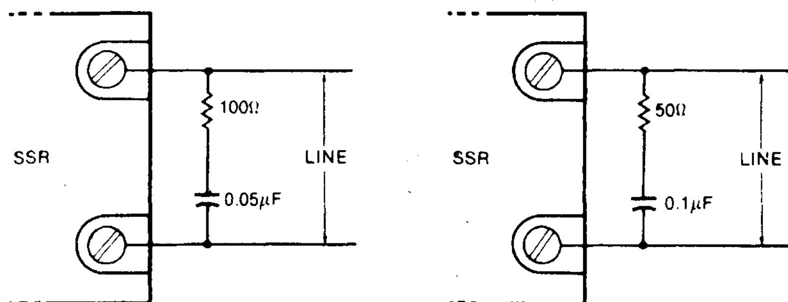


圖 8 —41 固態繼電器用的典型損減電路

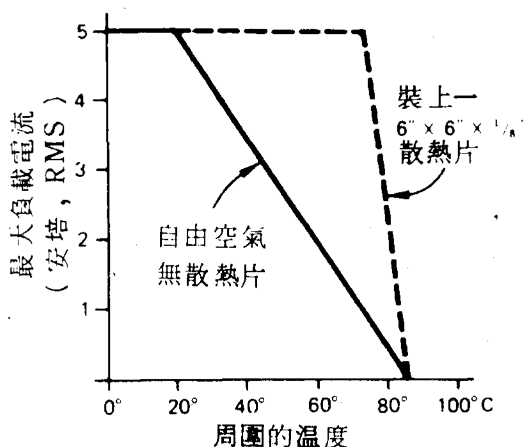


圖 8 —42 一固態繼電器之負載電流對溫度曲線

造成電壓降，產生一與繼電器所轉換之電流有直接關係的熱量。除非您以固態繼電器控制相當低電流的設備，否則，繼電器即應加有散熱片。圖 8 —42所示即爲一繼電器之溫度與

載流能力的典型關係圖。除非加有散熱裝置，否則，一般而言，一個 7 安培的固態繼電路將僅能承擔約 1.5 安培的負載。

固態繼電器在實際用於界面之前，還有一些其它考慮事項必須仔細加以評估。絕大多數繼電器在導通時都有一最低電流流過，而且控制邊與負載邊兩邊所加的電壓亦有一最高與最低極限。大多數情況下，您幾乎都會發現固態繼電器並非是萬用的，而且一個繼電器亦無法滿足您的所有需求。因此，若您希望以固態繼電器作界面，我們建議您與本章末了所列之繼電器製造商取得連繫，或詳查這些製造商所提供的規格表，使用說明等資料。

在結束固態繼電器以及其它用以控制一般稱爲非邏輯設備之元件的討論前，還有一種保護元件必須提一下。這種元件即爲金屬氧化變阻器，英文簡稱爲MOV。其可有各種不同的用法，可用以保護計算機、控制電路、界面、電源、以及計算機系統之其它重要部份，防止瞬間高電壓或脈衝的破壞干擾。此地，我們並不打算對此一元件作詳細深入的介紹，有興趣的讀者希望您逕行參考其它有關的資料，如本章最後所列之第 6 本參考書。從這本手册內，您可以找到更多有關如何保護您所投資購買之計算機系統以及其它電子儀器的資料。

喔，對了，我想有些讀者可能正在想，爲何到目前爲止，本章一直尙未談到標題上所抬列的電鈴與汽笛呢？事實上，電鈴就是一種簡單的繼電器電路罷了，不過，由於電鈴更常產生反電動勢，因此，使用電鈴時一定要記得使用良好的二極體保護電路。固態蜂鳴器可圓滿地解決這個問題，而且

其可以集極開路式晶片或週邊推動器推動，而無太多的紛擾。汽笛，如火警汽笛，一般均由一AC 117 伏特的馬達控制，這個馬達受一固態繼電器的控制，繼電器可由一集極開路式邏輯閘控制，而邏輯閘再進一步受計算機控制等等。

8－7 一些固態繼電器製造商

Douglas Randall
6 Pawcatuck Avenue
Pawcatuck, CT 02891

EI&S
42 Pleasant Street
Stomeham, MA 02180

Elec-trol, Inc.
26477 N. Golden Valley Rd.
Saugus, CA 91350

Gordos Arkansas, Inc.
1000 N. Second Street
Rogers, AR 72756

Gould, Inc.
Controls Division
100 Relay Road
Plantsville, CT 06479

Motorola Inc.
Subsystem Products
P. O. Box 29023
Phoenix, AZ 85038

Opto 22
5842 Research Drive
Huntington Beach, CA 92649

Sigma Instruments, Inc.
170 Pearl St.
Braintree, MA 02184

參考書

1. The Peripheral Driver Data Book, Texas Instruments, Inc., Dallas, TX 75222, 1977.
2. Linear and Interface Circuits Applications, Texas Instruments, Inc., Dallas, TX 75222, 1974.
3. Sprague Integrated Circuit Data Book (WR-500), Sprague Electric Co., Worcester, MA 01606, 1978.
4. Designer's Handbook of Solid-State Relays, Gordos Arkansas, Inc., Rogers, AR 72756, 1977.
5. Stepper Motor Handbook, North American Philips Controls Corp., Cheshire, CT 06510.
6. Transient Voltage Suppression Manual, 2nd ed., General Electric Co., Auburn, NY 13201, 1978.

附錄A

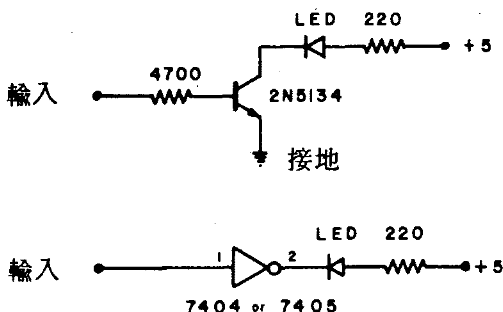
邏 輯 功 能

這本書的實驗用到許多邏輯功能。這些分別是燈炮顯視器（ lamp monitors ），邏輯開關，與脈衝電路。不論那一種，其等效電路均非常簡單，不過前面我們却僅一直使用其方塊圖，而非詳細電路圖。以下即對這幾個邏輯功能作更詳細的介紹。

燈炮顯視器

所謂燈炮顯視器就是發光二極體或其它能亮或暗的顯示元件，這種元件可顯示邏輯輸出的狀態。我們一向採用的規則是，邏輯 1 亮，邏輯 0 暗。圖 A － 1 所示之電路即可用以構成燈炮顯視器。

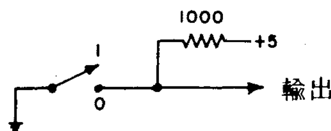
紅色的 LED 由於便宜且顯眼，因此，我們建議您採用紅色的ＬＥＤ。做書中的實驗時，您總共至少需要八個燈炮顯視器。



圖A－1　兩個可用於實驗中的簡易燈炮顯視器電路

邏輯開關

邏輯開關就是能供應實驗中所用之TTL吻合IC邏輯0或邏輯1電壓的開關。圖A－2所示即爲一典型的邏輯開關。這個開關可爲單刀單投的跳動開關或滑動開關均可。實驗至少需要八個邏輯開關。



圖A－2　能產生邏輯1或邏輯〇輸出的簡易邏輯開關電路圖

脈衝電路

實驗中所用的脈衝電路乃在產生一完全無"跳訊"（bounce）的"清徹"輸出。由於絕大多數開關均採用彈簧式之金屬接觸，因此，開關在打開或閉合之際，其接觸通常會打開或閉合數次。若以這種開關所產生的脈衝加給計數器，則開關每按一次，計數器很可能會數到 30 至 40 個脈衝不等。由於經常有許多場合都必須用到純粹的邏輯 1 至邏輯 0，或邏輯 0 至邏輯 1 轉換，因此，這時我們就必須使用經除跳訊過的開關。單刀雙投式的機械開關就很容易去除雜訊。圖 A－3 所示即爲一典型的除跳訊電路。這個電路以兩個 NAND 閘組成一可由開關設定或重置（清除）的正反器。電路有兩個輸出。當開關置於如圖所示的位置時，兩個邏輯閘的輸出即爲正當的邏輯狀態。當開關換至另一個位置時，NAND 閘的輸出將對調。脈衝電路最好均能用瞬間開關。

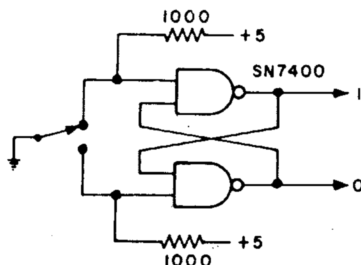


圖 A－3　以交錯連接之 NAND 閘去除接觸跳訊之脈衝電路

在做麵包板邏輯電路實驗時，燈炮顯視器，邏輯開關，與脈衝電路等都是非常有用的元件。以上三個電路雖然都很簡單，但有許多廠商都有做好，包含這三種電路以及其它邏輯功能的麵包板元件兜售。有興趣您可寫信到下列地址郵購

E & L Instruments, Inc.
61 First Street
Derby, CT 06418

AP Products, Inc.
Mentor, OH 44060

PACCOM
14825 NE 40th, Suite 340
Redmond,WA 98025

附錄B

實驗所需零件

4	SN7402	四倍NOR閘IC
2	SN7474	雙倍D型正反器IC
2	DM8095或SN74365	三態輸入緩衝器（每一輸入口兩個）
2	SN7475	四倍鎖住器IC
1	NE5018	八位元D／A轉換器IC（Signetics 公司）
1	SN7404	六倍反相器IC
2	SN74LS373	三態八倍鎖住器IC
1	0.01 μF	陶磁電容
1	4700歐姆，¼瓦電阻器	
6	220歐姆，¼瓦電阻器	
6	可見LED（紅色2個，綠色2個，黃色2個）	
1	10 K	電位計，可變型態
1	10 K，¼瓦電阻器	
1	100 μF	電解電容16WVDC
1	33 K，¼瓦電阻器	

1　150 PF　磁電容
1　2200 歐姆，¼瓦電阻器
1　ADC0804　類比至數位轉換器（國際半導體公司）
1　LM335　溫度察覺器
4　1000 歐姆，¼瓦電阻器

除了以上所列這些零件，在實驗 14 之邏輯測試器程式，您還會用到SN7400，SN7408，SN7402，SN7410，SN7486，SN7430，與SN7493 等各色各樣的 I C。作者希望您徹底的看看這個實驗，以得知您想測試的到底是什麼樣的電路。

其它用到的設備包括：一個 ± 12 伏特的電源（D／A 轉換器電路使用），掛鈎線，一個額外免焊的麵包板，脈衝電路，邏輯開關，燈炮顯示器，以及一電壓表或三用電表。

有關類比轉換器的資料可由下列機構獲得：

ADC0804 A/D Converter
National Semiconductor Corp.
2900 Semiconductor Drive
Santa Clara, CA 95051

NE5018 D/A Converter
Signetics Corporation
811 East Arques Avenue
Sunnyvale, CA 94086

各種 I C以及零件均有許多廠家製造供應，若有任何疑難，作者希望您多查查資料。我們已儘量地使用最標準的零件。

附錄C

6502微處理器技術資料

緊接後面幾頁爲有關6502 微處理器之技術資料。這些資料均取材自MOS Technology公司（6502 之製造商）所出版的"1980年零件資料目錄"。有關6502處理器之完整資料，以及其支援晶片之功能，讀者可參考許多其它有關之書籍。

6502晶片亦可由下面的兩家廠商買到：

Rockwell International
3310 Miraloma Avenue
Anaheim, CA 92803

Synertek, Inc.
3001 Stender Way
Santa Clara, CA 95051

這兩家廠商亦可提供您有關6502 微處理器晶片以及其它相關元件的資料。



MCS 6500微處理器族

一、特　色

- 單一＋5伏特電源
- N通道，矽閘，空負載技術
- 八位元並行處理
- 56個指令
- 十進與二進算術
- 並行結構
- 13種定址法
- 堆疊指示器可程式化，且堆疊器長度可變
- 可接任何型態或速度之記憶器
- 1或2 MHz 動作

二、說　明

MCS 6500系列微處理器代表一個在軟體上完全吻合的微處理器族。這個產品族包括許多在軟體上完全相通，但在可選取之記憶空間、插斷輸入與內含時序振盪器及推動器上卻有多種選擇之微處理器。所有MCS 6500族內的微處理器，在軟體上均完全通用，且其巴士與M 6800之產品相配。

MCS 6500系列包含五個由內含時序振盪器與推動器推動的微處理器，以及四個由外加之時序推動的微處理器。內含時序類的主要目標是以單相輸入、晶體或RC輸入產生時基的高性能廉價應用。外加時序類主要則針對最大時序控制毫無選擇餘地的多處理器系統應用。每一類微處理器均分別有最高工作頻率1 MHz與2 MHz（產品編號加一個A）的兩種。

三、族成員

編號		時序	接脚數	IRQ	NMI	RDY	選取能力
塑膠	陶瓷						
MCS6502	MCS6502	內含	40	√	√	√	16(64K)
MCS6503	MCS6503	〃	28	√	√		12(4K)
MCS6504	MCS6504	〃	28	√			13(8K)
MCS6505	MCS6505	〃	28	√		√	12(4K)
MCS6506	MCS6506	〃	28	√			12(4K)
MCS6507	MCS6507	〃	28			√	13(8K)
MCS6512	MCS6512	外加	40	√	√	√	16(64K)
MCS6513	MCS6513	〃	28	√	√		12(4K)
MCS6514	MCS6514	〃	28	√			13(8K)
MCS6515	MCS6515		28	√		√	12(4K)

四、接脚功能

時序(ϕ_1與ϕ_2)

MCS651X必須使用雙相、電壓準位等於V_{cc}之非重疊時序。

MCS 650X 之時序則由內含之時序產生器供應。這些時序的頻率由外部控制。詳情請見MCS 6502的這項資料說明。

位址巴士(A0-A15)

（見每一微處理器之位址線部份）

這些輸出均爲TTL吻合，能推動一個標準的TTL負載與130PF。

資料巴士(D0-D7)

資料巴士使用八支接脚。此一巴士爲雙向性巴士，傳遞進出設備與週邊的資料。輸出爲三態緩衝，能推動一個標準的TTL負載與130 PF。

資料巴士致能(DBE)

這個TTL 吻合的輸入使三態的資料輸出緩衝器能受外部的控制，接高電位，其令微處理器之巴士推動器致能。正常作業時，DBE由第2相時序（ϕ_2)推動，因此，微處理器之資料僅能於ϕ_2時輸入。在讀取期間，資料巴士推動器內部禁能，變成開路。欲由外部令資料巴士推動器禁能時，DBE應接低電位。

準備好(RDY)

這個輸入信號使用者能令微處理器一個時序週期一個時序週期地進行除了寫入週期以外的一切週期。若與第一相時序（ϕ_1）同時轉換至低電位狀態，則這個信號將令微處理器停止於輸出位址線上呈現目前正在拿取之位置位址的狀態。這種狀態將一直延續過緊接之第2相時序（ϕ_2）。這個特色

使微處理器能與慢速的PROM與快速（最快2 MHz）的DMA界面。若RDY 信號變低電位時正好爲寫入週期，則其將暫時被忽略，直至下個讀取作業時，才被考慮。

插斷請求($\overline{IRQ}$)

這個與TTL 吻合的輸入信號，要求微處理器進入一個插斷系列。微處理器會先執行完目前正在執行的指令，然後才認知這個請求。此時，微處理器會檢查狀態暫存器內的插斷罩蓋（ interrupt mask ）位元。若此一插斷罩蓋位元之值爲0 ，則微處理器就開始執行插斷系列。執行插斷系列時，微處理器先將程式計數器與狀態暫存器的內含存入堆疊器（ stack ）。接著將插斷罩蓋位元設定爲1 ，使插斷不再重複發生。在這個週期終了時，位址FFFE 的記憶位置內含取入程式計數器之低次位元組，且位址FFFF 之記憶位置內含取入程式計數器之高位元組，令控制轉移至這個位址向量所指的記憶位置上。插斷欲被認可，RDY 信號必須處於高電位狀態。您必須以一外部的3 KΩ 電阻作適當的“結合OR ”（ wire-OR ）運算。

不可罩蓋插斷($\overline{NMI}$)

這個輸入信號的負向轉換要求微處理器產生一不可罩蓋（即不可忽略）之插斷系列。

$\overline{NMI}$ 爲一無條件插斷。在完成目前正在執行的指令後，不論插斷罩蓋位元之狀態如何，微處理將進行剛剛在$\overline{IRQ}$段落內所提的插斷起始系列。不過，這回取入程式計數器者

爲分別來自位址FFFA(低)與FFFB(高)兩記憶位置之內含，而非來自FFFE與FFFF 之內含。這些位置上所存的指令將令微處理器跳至一非罩蓋的插斷常式上。$\overline{\text{NMI}}$ 亦須以一接至Vcc 的外加3KΩ電阻，作適當的結合OR運算。

$\overline{\text{IRQ}}$ 與 $\overline{\text{NMI}}$ 輸入均爲硬體插斷線，微處理均在 ϕ_2 時讀取這兩個輸入的狀態，並於完成指令執行後之 ϕ_1 期間，開始適當的插斷常式。

置定溢位旗號(S.O.)

這個輸入的負向緣將狀態暫存器中的溢位(overflow)旗號值設定爲1 。微處理器於 ϕ_1 的後緣時查看這個信號。

SYNC

這個輸出線主要在揭示微處理器正在作運算碼拿取 (OP code fetch)的週期。SYNC線於運算碼拿取的 ϕ_1 期間變成高電位，並一直保持至拿取週期結束爲止。若RDY 線在SYNC變高電位的 ϕ_1 時序脈衝期間接低電位，則微處理器將停在其現有狀態，直至RDY 線變高電位爲止。依此，您即可以SYNC 信號控制RDY ，令微處理器作單指令的執行——一次僅執行一個指令。

重置(Reset)

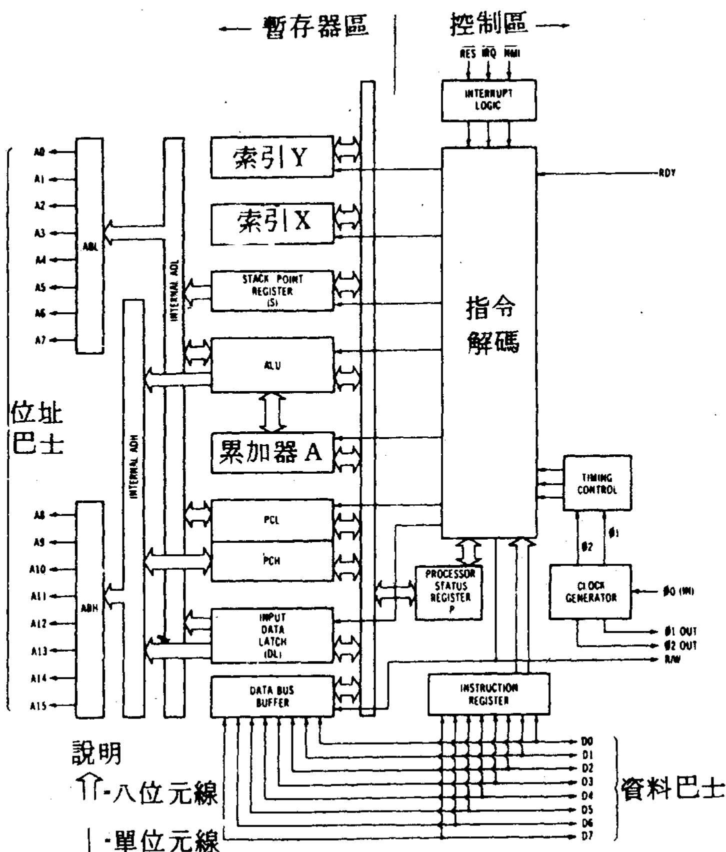
這個輸入令微處理重新開始。這條線保持低電位期間，一切讀／寫微處理器的動作均屬無效。一測及這個輸入的正向緣，微處理器即立即開始重置系列。

在六個時序週期的系統起始時間過後，插斷罩蓋旗號將被設定爲1，且微處理器將位址FFFC與FFFD 兩記憶位置的內含取入程式計數器。這兩個位置之內含所指的，就是控制程式之最開始位置。

開機過程中，當Vcc 抵達4.75 伏特後，重置線之低電位至少必須再保持兩個時序週期的期間。這時候，R／W與(SYNC)信號將變成有效。

這兩個時序週期過後，當重置信號變爲高電位時，微處理器將進行以上所述的正常重置程序。

五、內部構造



6502 微處理器內部構造之方塊圖

附註：

1. MCS 6512，13，14，15 等並不含時序產生器。
2. 每一個MCS 65 XX的選址能力與控制信號均有所差異。

六、指令集—依英文字母順序排列

ADC	Add to Accumulator with Carry
AND	AND Memory with Accumulator
ASL	Accumulator Shift Left
BCC	Branch on Carry Clear
BCS	Branch on Carry Set
BEQ	Branch on Result Equal to Zero
BIT	Test Bits in Memory with Accumulator
BMI	Branch on Result Minus
BNE	Branch on Result Not Equal to Zero
BPL	Branch on Result Plus
BRK	Force Break
BVC	Branch on Overflow Clear
BVS	Branch on Overflow Set
CLC	Clear Carry Flag
CLD	Clear Decimal Mode
CLI	Clear Interrupt Disable Bit
CLV	Clear Overflow Flag
CMP	Compare Memory and Accumulator
CPX	Compare Memory and Index X
CPY	Compare Memory and Index Y
DEC	Decrement Memory by One
DEX	Decrement Index X by One
DEY	Decrement Index Y by One
EOR	Exclusive-OR Memory with Accumulator
INC	Increment Memory by One
INX	Increment Index X by One
INY	Increment Index Y by One
JMP	Jump
JSR	Jump to Subroutine
LDA	Load Accumulator with Memory
LDX	Load Index X with Memory
LDY	Load Index Y with Memory
LSR	Logical Shift Right
NOP	No Operation
ORA	OR Memory with Accumulator
PHA	Push Accumulator on Stack
PHP	Push Processor Status on Stack
PLA	Pull Accumulator from Stack
PLP	Pull Processor from Stack
ROL	Rotate Left
ROR	Rotate Right
RTI	Return from Interrupt
RTS	Return from Subroutine
SBC	Subtract from Accumulator with Carry
SEC	Set Carry Flag
SED	Set Decimal Mode
SEI	Set Interrupt Disable Status
STA	Store Accumulator in Memory
STX	Store Index X in Memory
STY	Store Index Y in Memory
TAX	Transfer Accumulator to Index X
TAY	Transfer Accumulator to Index Y
TSX	Transfer Stack Pointer to Index X
TXA	Transfer Index X to Accumulator
TXS	Transfer Index X to Stack Pointer
TYA	Transfer Index Y to Accumulator

七、定址法(Addreessing Modes)

累加器定址

這種定址的指令均長一個位元組，指令對累加器之內含作運算。

立即定址

立即定址時，指令之第二個位元組即爲運算元（ operand ），因此，指令不需再作進一步記憶器選取。

絕對定址

絕對定址指令長三個位元組，第二位元組爲運算元位址之低次八位元，第三位元組爲運算元位址之高次八位元。由此可見，絕對定址可存取 65 K 個記憶位置。

零頁定址

零頁定址的著眼乃在縮短指令，並節省指令執行時間。指令長兩個位元組，第二位元組爲運算元位址之低次八位元，運算元位址之高次八位元則假設均爲零。細心使用零頁定址將可大幅提高程式的效率。

索引零頁定址(X , Y索引)

這種定址方式與索引暫存器合用，並分別稱爲"零頁，X"或"零頁，Y"。有效位址乃由指令之第二位元組與索

引暫存器內含相加求得。由於這亦是"零頁"定址的一種，因此，指令之第二位元組選取一零頁位置。另外，由於"零頁"的特性，故沒有進位加至記憶位址之高次位元組，且永遠不會有超越頁界的情形發生。

索引絕對定址（X，Y索引）

這種定址與X及Y暫存器合用，且分別稱爲"絕對，X"與"絕對，Y"。運算元之有效位址由X或Y之內含與指令之二、三位元組相加求得。這種方式等於指令含基底位址，而索引暫存器含位移或索引值。這種方式使得一組少數幾個指令能存取許多個記憶位置，減少了程式寫碼與執行時間。

隱含定址

隱含定址即運算元位址隱含於指令運算碼內的定址法。

相對定址

相對定址僅用於跳越（ jump ）指令，以形成條件跳越之目的地。指令執行時，指令第二位元組變成運算元，被加至程式計數器之低次位元組，產生下一個欲執行之指令的指示器。由次一緊接指令算起，這種跳越的範圍僅及往前128，往後127個位元組。

索引間接定址

索引間接定址，記爲（間接，X），將指令之第二位元組加至X索引暫存器，捨棄進位。然後，加算所得結果所指

之零頁位置的內含即爲有效位址之低次八位元，而次一緊接零頁位置所含即爲有效位址之高次八位元。含有效位址的兩個記憶位置都必須在第零頁內。

間接索引定址

於間接索引定址，指令之第二位元組指至一零頁位置。該零頁位置之內含與Y索引暫存器之內含相加，所得當爲有效位址之低次八位元。然後加算之進位再與前述零頁位置之次一隣接位置的內含相加，所得當成有效位址之高次八位元。

絕對間接定址

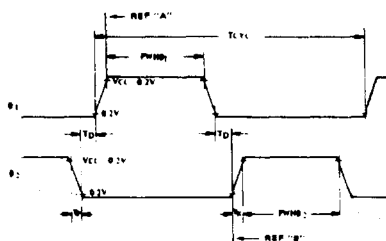
指令之第二（當低次）與第三（當高次）位元組所指之記憶位置，含有效位址之低次八位元。次一緊接位置含有效位址之高次八位元。有效位址被存入十六位元之程式計數器。

八、最大額定值

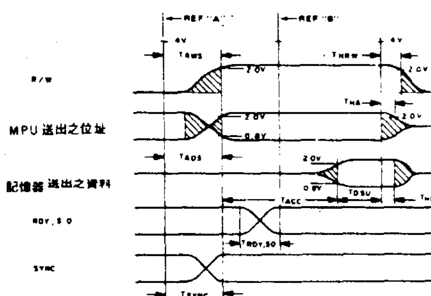
項目	符號	額定值	單位
電源電壓	V_{CC}	－0.3至＋7.0	V_{dc}
輸入電壓	V_{IN}	－0.3至＋7.0	V_{dc}
動作溫度	T_A	0至70	℃
儲存溫度	T_{STG}	－55至＋150	℃

注　意：这个元件含有由于高输入静电或电场所造成之損害的保护，不过，使用时加电压还是要特别小心，切勿加上超过最大额定的电压。

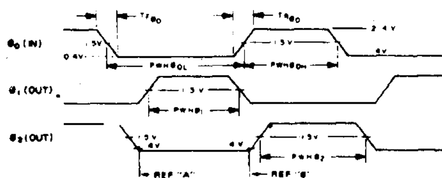
時序信號 —MCS6512, 13, 14, 15



自記憶器或週邊讀取資料的時序



資料寫入記憶器或週邊的時序

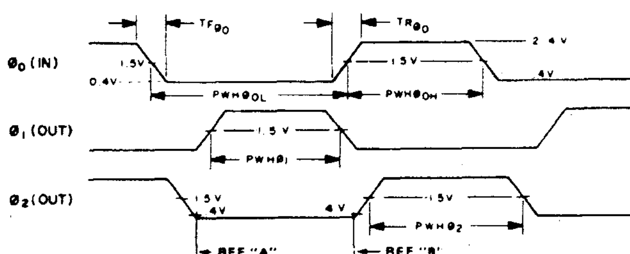


NOTE
"REF" 指時序參考點

電機特性　(V_{CC} = 5.0V ± 5%, V_{SS} = 0, T_A = [illegible]°C)
$\emptyset_1$, $\emptyset_2$ 適合 MC56512, 13, [illegible], 15, $\emptyset_{o(in)}$ 適合 MC56502, 03, 04, 05 and 06

符號	參數	最小	Typ	最大	單位	測試狀況
V_{IH}	輸入高電壓	V_{SS} + 2.4 V_{CC} − 0.2		V_{CC} V_{CC} + 0.25	Vdc	Logic, $\emptyset_{o(in)}$ $\emptyset_1$, $\emptyset_2$
V_{IL}	輸入低電壓	V_{SS} − 0.3 V_{SS} − 0.3		V_{SS} + 0.4 V_{SS} + 0.2	Vdc	Logic, $\emptyset_{o(in)}$ $\emptyset_1$, $\emptyset_2$
V_{IHT}	輸入高 Threshold Voltage	V_{SS} + 2.0			Vdc	$\overline{RES}$, $\overline{NMI}$, RDY, $\overline{IRQ}$, Data, S.O
V_{ILT}	Input Low Threshold Voltage			V_{SS} + 0.8	Vdc	$\overline{RES}$, $\overline{NMI}$, RDY, $\overline{IRQ}$, Data, S.O
I_{IN}	輸入漏電流			2.5 100 10.0	μA μA μA	(V_{IN} = 0 to 5.25V, V_{CC} = 0) Logic (Excl. RDY, S.O.) $\emptyset_1$, $\emptyset_2$ $\emptyset_{o(in)}$
I_{TSI}	三態（off 態）輸入電流			10	μA	(V_{IN} = 0.4 to 2.4V, V_{CC} = 5.25V) Data Lines
V_{OH}	輸出高電壓	V_{SS} + 2.4			Vdc	(I_{LOAD} = −100μAdc, V_{CC} = 4.75V) SYNC, Data, A0-A15, R/W
V_{OL}	輸出低電壓			V_{SS} + 0.4	Vdc	(I_{LOAD} = 1.6mAdc, V_{CC} = 4.75V) SYNC, Data, A0-A15, R/W
P_D	功率消耗		.25	.70	W	
C C_{IN} C_{OUT} $C_{\emptyset o(in)}$ $C_{\emptyset 1}$ $C_{\emptyset 2}$	電容量		 30 50	 10 15 12 50 50 80	pF	(V_{IN} = 0, T_A = 25°C, f = 1MHz) Logic Data A0-A15, R/W, SYNC $\emptyset_{o(in)}$ $\emptyset_1$ $\emptyset_2$

附註：
$\overline{IRQ}$ and $\overline{NMI}$ 需 3K 之提升電阻



1 MHz TIMING

CLOCK TIMING—MCS6512, 13, 14, 15

符號	特　　性	最小	Typ	最大	單位
T_{CYC}	Cycle Time	1000			nsec
PWH ϕ1 PWH ϕ2	Clock Pulse Width ϕ1 (Measured at V_{CC} − 0.2 V) ϕ2	430 470			nsec
T_F	Fall Time (Measured from 0.2 V to V_{CC} − 0.2 V)			25	nsec
T_D	Delay Time Between Clocks (Measured at 0.2 V)	0			nsec

CLOCK TIMING—MCS6502, 03, 04, 05, 06

符號	特　　性	最小	Typ	最大	單位
T_{CYC}	Cycle Time	1000			ns
$PWH\phi_o$	$\phi_{o\,(IN)}$ Pulse Width (measured at 1.5 V)	460		520	ns
$TR\phi_o$, $TF\phi_o$	$\phi_{o\,(IN)}$ Rise, Fall Time			10	ns
T_D	Delay Time Between Clocks (measured at 1.5 V)	5			ns
$PWH\phi_1$	$\phi_{1\,(OUT)}$ Pulse Width (measured at 1.5 V)	$PWH\phi_{oL}$ − 20		$PWH\phi_{oL}$	ns
$PWH\phi_2$	$\phi_{1\,(OUT)}$ Pulse Width (measured at 1.5 V)	$PWH\phi_{oH}$ − 40		$PWH\phi_{oH}$ − 10	ns
T_R, T_F	$\phi_{1\,(OUT)}$, $\phi_{2\,(OUT)}$ Rise, Fall Time (measured .8 V to 2.0 V) (Load = 30pF + 1 TTL)			25	ns

讀取／寫入時序

符號	特　　性	最小	典型	最大	單位
T_{RWS}	Read/Write Setup Time From MCS6500		100	300	ns
T_{ADS}	Address Setup Time From MCS6500		100	300	ns
T_{ACC}	Memory Read Access Time			575	ns
T_{DSU}	Data Stability Time Period	100			ns
T_{HR}	Data Hold Time – Read	10			ns
T_{HW}	Data Hold Time – Write	30	60		ns
T_{MDS}	Data Setup Time From MCS6500		150	200	ns
T_{RDY}	RDY, S.O. Setup Time	100			ns
T_{SYNC}	SYNC Setup Time From MCS6500			350	ns
T_{HA}	Address Hold Time	30	60		ns
T_{HRW}	R/W Hold Time	30	60		ns

2 MHz 時序

CLOCK TIMING—MCS6512, 13, 14, 15, 16

符號	特性	最小	Typ	最大	單位
T_{CYC}	Cycle Time	500			nsec
PWH $\phi 1$ PWH $\phi 2$	Clock Pulse Width $\phi 1$ (Measured at $V_{CC} - 0.2$ V) $\phi 2$	215 235			nsec
T_F	Fall Time (Measured from 0.2 V to $V_{CC} - 0.2$ V)			12	nsec
T_D	Delay Time Between Clocks (Measured at 0.2 V)	0			nsec

CLOCK TIMING—MCS6502, 03, 04, 05, 06

符號	特性	最小	Typ	最大	單位
T_{CYC}	Cycle Time	500			ns
$PWH\phi_O$	$\phi_{O(IN)}$ Pulse Width (measured at 1.5 V)	240		260	ns
$TR\phi_O$, $TF\phi_O$	$\phi_{O(IN)}$ Rise, Fall Time			10	ns
T_D	Delay Time Between Clocks (measured at 1.5 V)	5			ns
$PWH\phi_1$	$\phi_{1(OUT)}$ Pulse Width (measured at 1.5 V)	$PWH\phi_{oL}-20$		$PWH\phi_{oL}$	ns
$PWH\phi_2$	$\phi_{2(OUT)}$ Pulse Width (measured at 1.5 V)	$PWH\phi_{oH}-40$		$PWH\phi_{oH}-10$	ns
T_R, T_F	$\phi_{1(OUT)}$, $\phi_{2(OUT)}$ Rise, Fall Time (measured .8 V to 2.0 V) (Load = 30pF + 1 TTL)			25	ns

READ/WRITE TIMING

符號	特性	最小	Typ	最大	單位
T_{RWS}	Read/Write Setup Time From MCS6500A		100	150	ns
T_{ADS}	Address Setup Time From MCS6500A		100	150	ns
T_{ACC}	Memory Read Access Time			300	ns
T_{DSU}	Data Stability Time Period	50			ns
T_{HR}	Data Hold Time – Read	10			ns
T_{HW}	Data Hold Time – Write	30	60		ns
T_{MDS}	Data Setup Time From MCS6500A		75	100	ns
T_{RDY}	RDY, S.O. Setup Time	50			ns
T_{SYNC}	SYNC Setup Time From MCS6500A			175	ns
T_{HA}	Address Hold Time	30	60		ns
T_{HRW}	R/W Hold Time	30	60		ns

附錄D

Apple界面麵包板零件

組成Apple 界面麵包板所需的零件爲：

IC 1 & 7	16支接脚的電阻網路，8個獨立的1000歐姆電阻
IC 2 & 6	8點的DIP開關（on-off）
IC 3，4，& 5	SN74LS85四倍比較器IC（切勿以SN74L85代替）
IC 8	SN74LS20雙倍四輸入NAND閘IC
IC 9	SN74365或DM8095三態緩衝器
IC 10 & 11	8216非反相巴士緩衝器，Intel或等效品
IC 12	SN74154解碼器IC
IC 13	SN7404反相器IC
IC 14	SN74123或SN74SL123雙倍單穩定IC
IC 15	LM319N雙倍比較器（14脚包裝）
IC 16，17，18 & 20	高品質16脚IC插座，Augat 516－AG－10D或等效品
IC 19	高品質8脚IC插座，Augat 508－AG－

	10 D或等效品
D1 − D4	1N4001 50 Piv，1安培二極體*
D5	黃色LED
D6	紅色LED
D7	綠色LED
D8 & D9	1N4148或1N4154，小信號二極體。
R1 & R8	1000歐姆，¼瓦電阻器
R2 & R3	220歐姆，¼瓦電阻器
R4 & R5	47K，¼瓦電阻器
R6	3900歐姆，¼瓦電阻器
R7	2200歐姆，¼瓦電阻器
C1	2200μF，16Vdcw電解電容（軸形）*
C2，4，& 5	0.1μF陶瓷，50伏特電容器
C3 & C6	1μF，35伏特dcw鉭電解電容器
C7 & C8	3.3μF，50伏特dcw電解電容器（軸形）
VR	LM309K 5伏特，1安培電壓調節器*
P1	Molex直角6脚接點（PN 09−75−1061）可有可無 需1個相配的母體（PN 09−50−7061）以及6個接脚（PN08−50−0106或08−50−0108）
P2	40脚直角跳線頭，AP產品923875R或等效
T1	12.6伏特之交流變壓器，1安培
Misc	11個　16脚IC插座

3 個　14 脚 IC 插座

1 個　24 脚 IC 插座

電纜組合：一端為 40 脚之接頭，另一端為 40 脚之板邊緣接點，面向同一方向。

免焊麵包板插座，SK-10，Superstrip或等效品，4 個 4－40×5/8之平頭螺絲，4 個 # 4 內齒橡皮鎖墊，4 個 # 4 六角形螺絲帽。

VR 散墊片，2 個 4－40×1/2螺絲，2 個 # 4 內齒橡皮鎖墊，2 個 # 4 六角螺絲帽，雲母絕緣片，散熱油（可有可無）。

電源線

若系統以外加之＋5 伏特電源作電源供給，則標有"*"號的零件就不必了。

包括電源在內的整套成品，可由下面之住址購得：

E & L Instruments, Inc.
61 First Street
Derby, CT 06418

附錄E

印刷電路板

圖 樣

這個附錄含有Apple 界面麵包板之印刷電路板的設計圖樣。實際使用時，這些圖案必須再經放大。我們建議您請一家照相館作出高對比的底片。每一圖形中之長黑線應放大至4英吋長。放大過程必須精準控制至所得底片正好與印刷底板同大小。零件輪廓圖您可能用不上，不過，其却可作您擺置各種零件的參考。

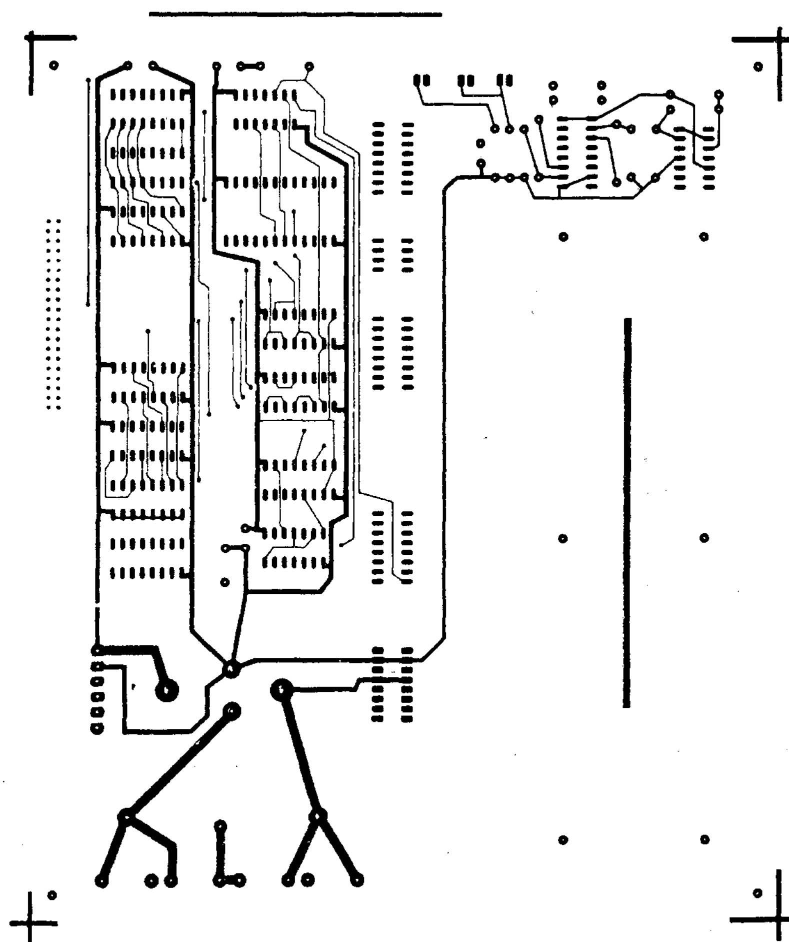


圖 E－1　界面麵包板之零件邊的印刷電路板圖樣(從右邊看)

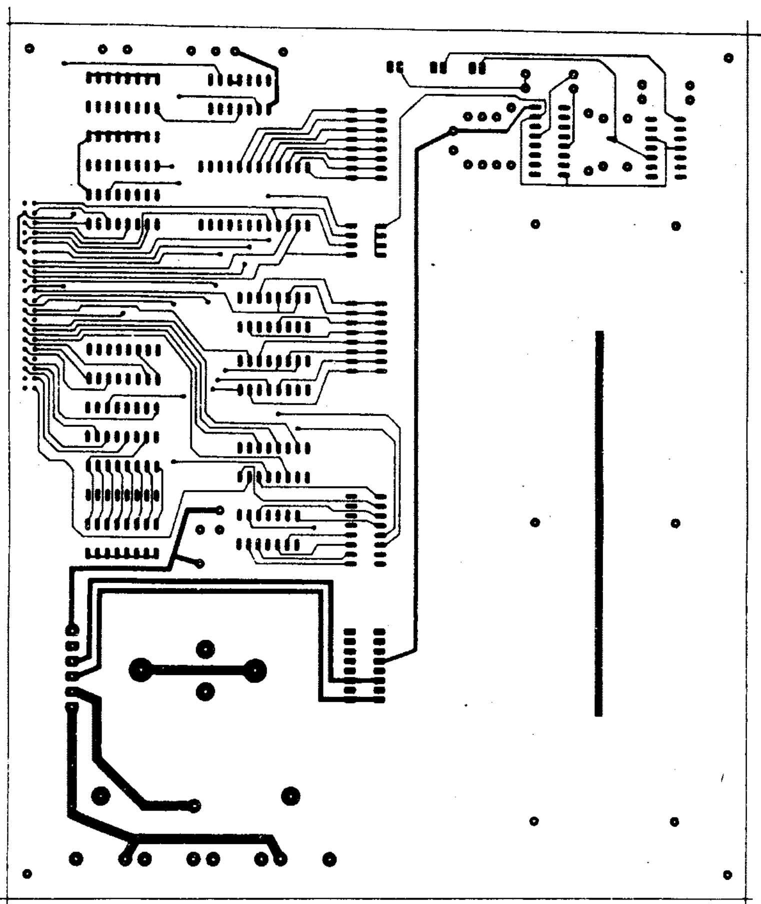
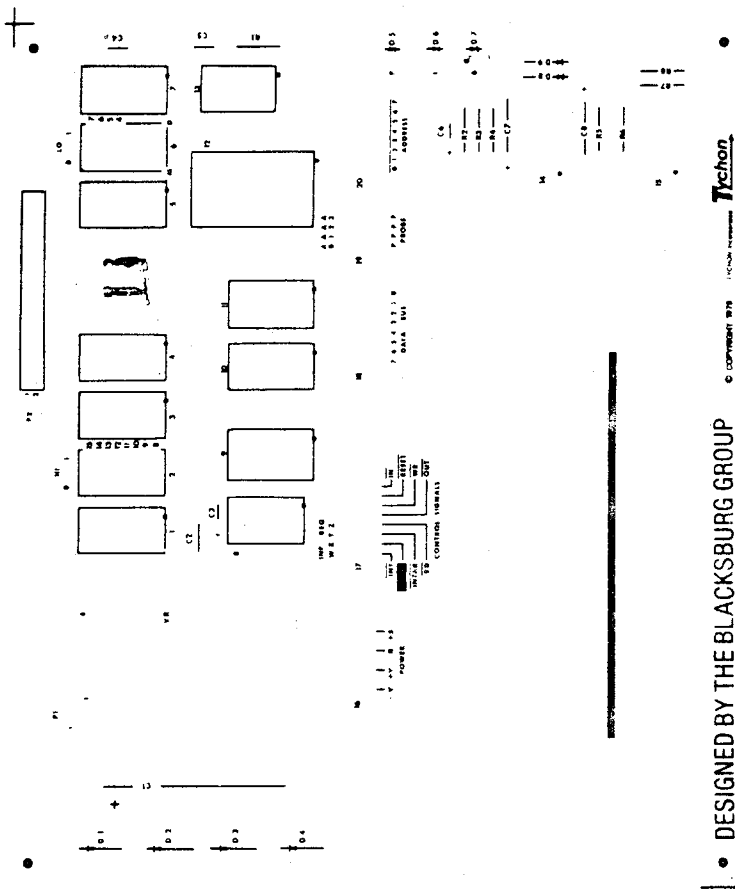


圖 E－2　界面麵包板焊點側之印刷電路板圖樣（反過來看）



圖E－3　界面麵包板之零件位置圖(由右邊看)

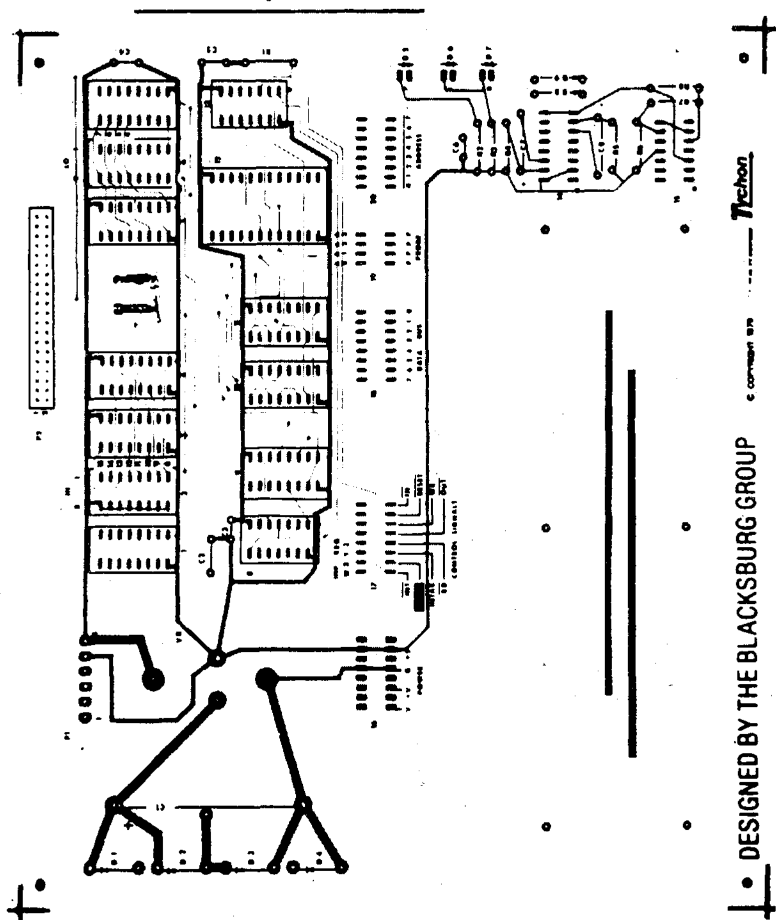


圖 E－4　零件裝上後之錫箔圖樣，可作裝插零件的參考

附錄F

在您的 Apple Ⅱ 上加上PIA

假若您希望在您的Apple 計算機上增加一個並行口，則您可在您的計算機上接上一個6520 PIA（ Peripheral Interface Adapter ，週邊界面調適器），這個結果即如圖F－1所示。

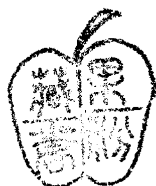
在這個界面內，PIA之低電位動作選擇線 $\overline{CS2}$ 永遠接地，且高電位動作之選擇線CS0與CS1，經一反相器接至低電位動作之設備選擇線 $\overline{\text{DEVICE SELECT}}$（由 Apple Ⅱ之電路所產生）。這個信號在微處理器選取某一個特定的記憶區時，選擇了特別的擴充槽。$\overline{\text{DEVICE SELECT}}$ 信號大大地簡化了界面工作。

這個界面很容易達成。最初的原形製作在一免焊的原形板上，界面槽接點信號以帶電纜拉出在16接脚的DIP（雙列並行包裝）接點上。然後，這些接點再插入已接線好，準備將擴充槽信號連接至16脚DIP插座之Apple Ⅱ 擴充槽模板（ Vector 4609 DP或同等品）上。PIA之PA0至PA7以及PB0至PB7以－25 對的電纜拉出至“外界”。界面的擺設並無多大關係。

為了測試這個界面，我們將一個DIP開關組與提升電阻接至PIA之PA0至PA7，同時，PB0至PB7亦經7404反相緩衝器（見圖F－2）與降壓電阻接至LED。然後，將例題F－1所示的組合語言程式，利用Apple II的迷你組譯程式（miniassembler）打入計算機，存於位址0300 H的位置，並教計算機開始執行。程式將A口設定成輸入且將B口設定成輸出。因此，PB0至PB7一開始均設定成邏輯0狀態。

程式不斷地讀取A口之內含，並將之寫至B口。此舉等於將A口之每一輸入開關的狀態傳送至其相對之LED(LED加邏輯1時發亮）。設意地改變DIP開關的設定，您即可測試到每一條線以及至Apple II的界面。（事實上，這整個測試程序就等於以計算機將開關與LED之間連成一線。）

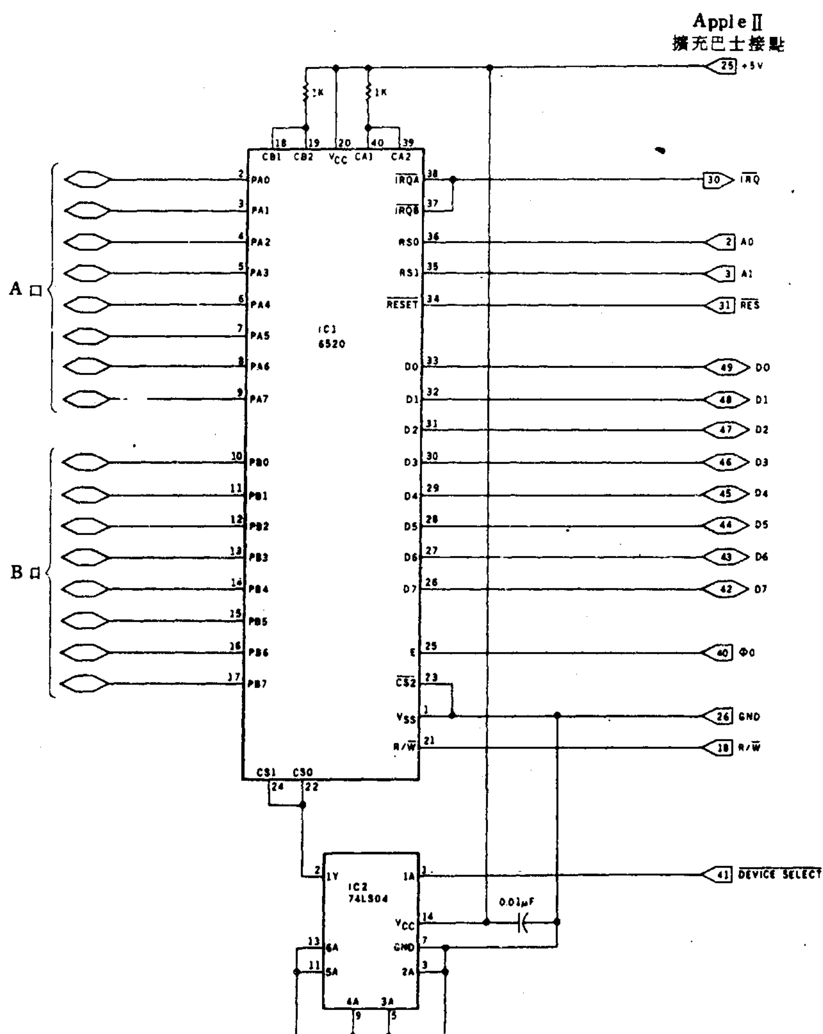
這個界面有一個缺點（見圖F－1），是PIA並未完全且獨特地解碼；換言之，指定給擴充槽之許多其它位址亦可選取此一PIA。圖F－3所示即為一種克服此一問題的方法。在這個電路中，74LS 42解碼器解碼A 2與A 3位址線，致使PIA僅佔用16個指定給Apple II擴展槽之位址中的4個位址。這樣一來，您又可在同一板上加上另一個PIA。



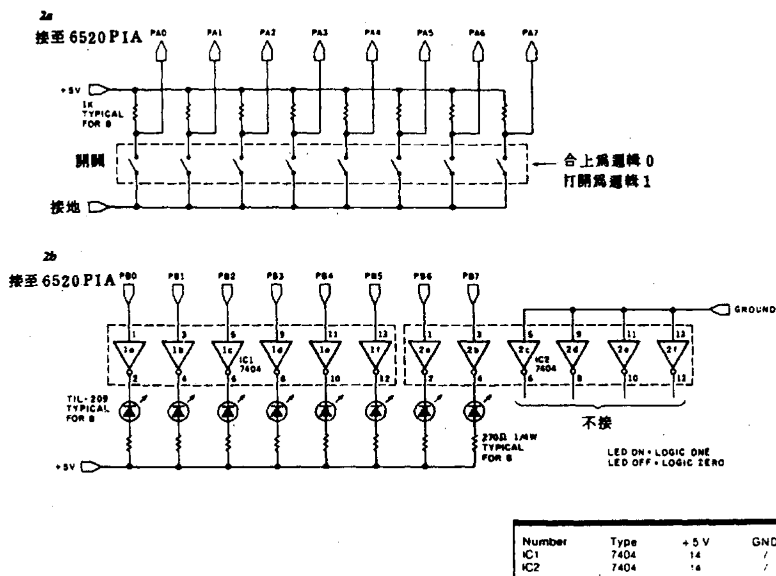
例題 F－1：測試6502界面的程式

```
; 這個程式自 6520 PIA之A口讀入開關的設定值，
; 並將之立即顯示於B口之LED上 。執行程式前以及欲結
; 束程式執行時，請將Apple II Reset。
; 這個程式所用的4號擴充槽之PIA暫存器位址為：
; $C0C0 =資料方向暫存器A（DDRA）／輸出暫存器A
;            （ORA）。
; $C0C1 =控制暫存器A（CRA）。
; $C0C2 =資料方向暫存器B（DDRB）／輸出暫存器B
;            （ORB）。
; $C0C3=控制暫存器B（CRB）。

$0300   LDA   #$FF
$0302   STA   $C0C2    ；設定PB0～PB7為輸出。
$0305   LDA   #$04
$0307   STA   $C0C1    ；將ORA致能，DDRA禁能。
$030A   STA   $C0C3    ；將ORB致能，DDRB禁能。
$030D   LDA   $C0C0    ；開關狀態讀入累加器。
$0310   STA   $C0C2    ；然後寫出至PB0～PB7。
$0313   JMP   $030D    ；一直反覆至Reset為止。
```

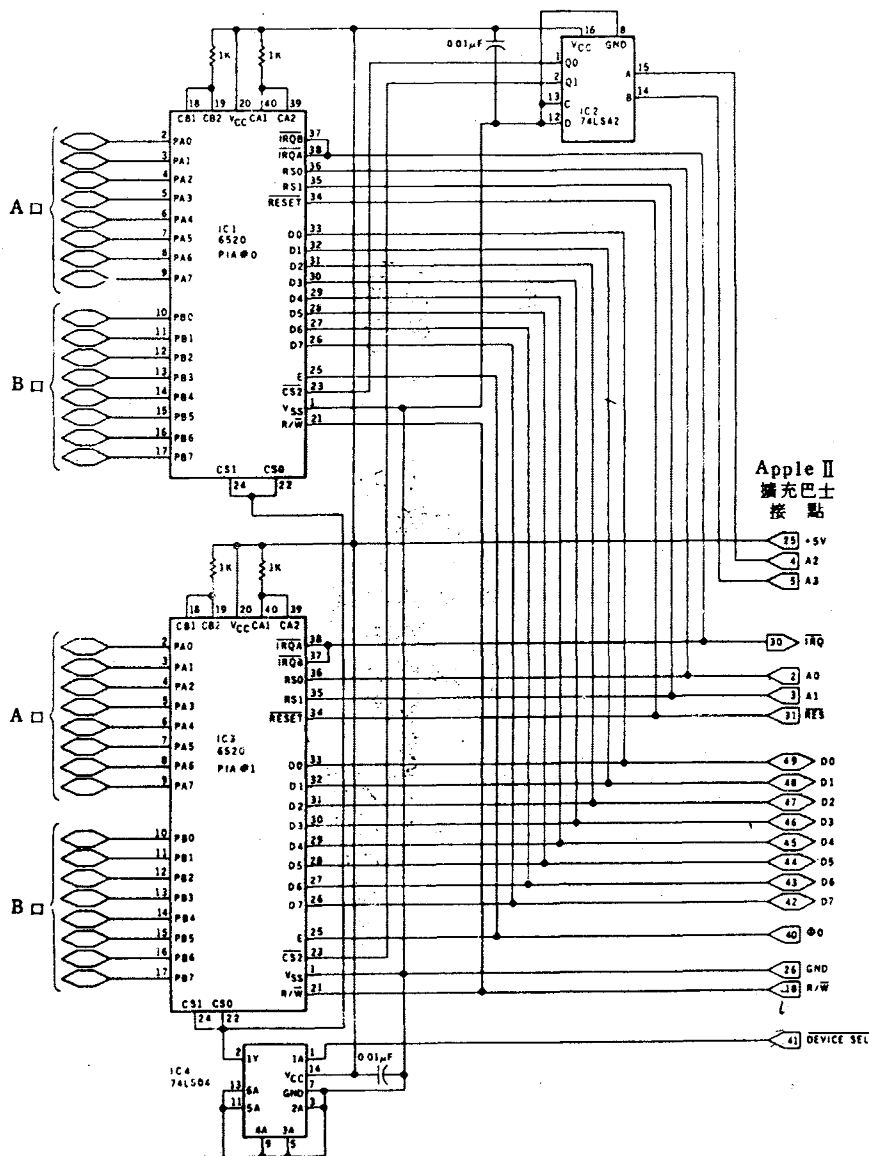


圖F－1　將Apple Ⅱ界面至一 6520 PIA。6520 之低電位動作選擇線接地，而另兩個高電位動作選擇線同時經一反相器接至$\overline{\text{DEVICE SELECT}}$(Apple Ⅱ產生，將其八個週邊位置其中之一致能的信號)。



Number	Type	+5 V	GND
IC1	7404	14	7
IC2	7404	14	7

圖 F － 2　測試圖 F － 1 之電路的控制開關與 LED 顯示器。圖 F － 2 a 所示爲以手設定 6520 之一口的邏輯狀態情形。圖 F － 2b 所示則爲顯示 6520 之一口上每一位元之邏輯狀態的情形。



圖F－3　加上一74LS42解碼器，使一個Apple II 擴充口能選取（即連接）一個以上的6520PIA。

内部交流

G16/3 APPLE界面实验
(中3—5/45)

D00200